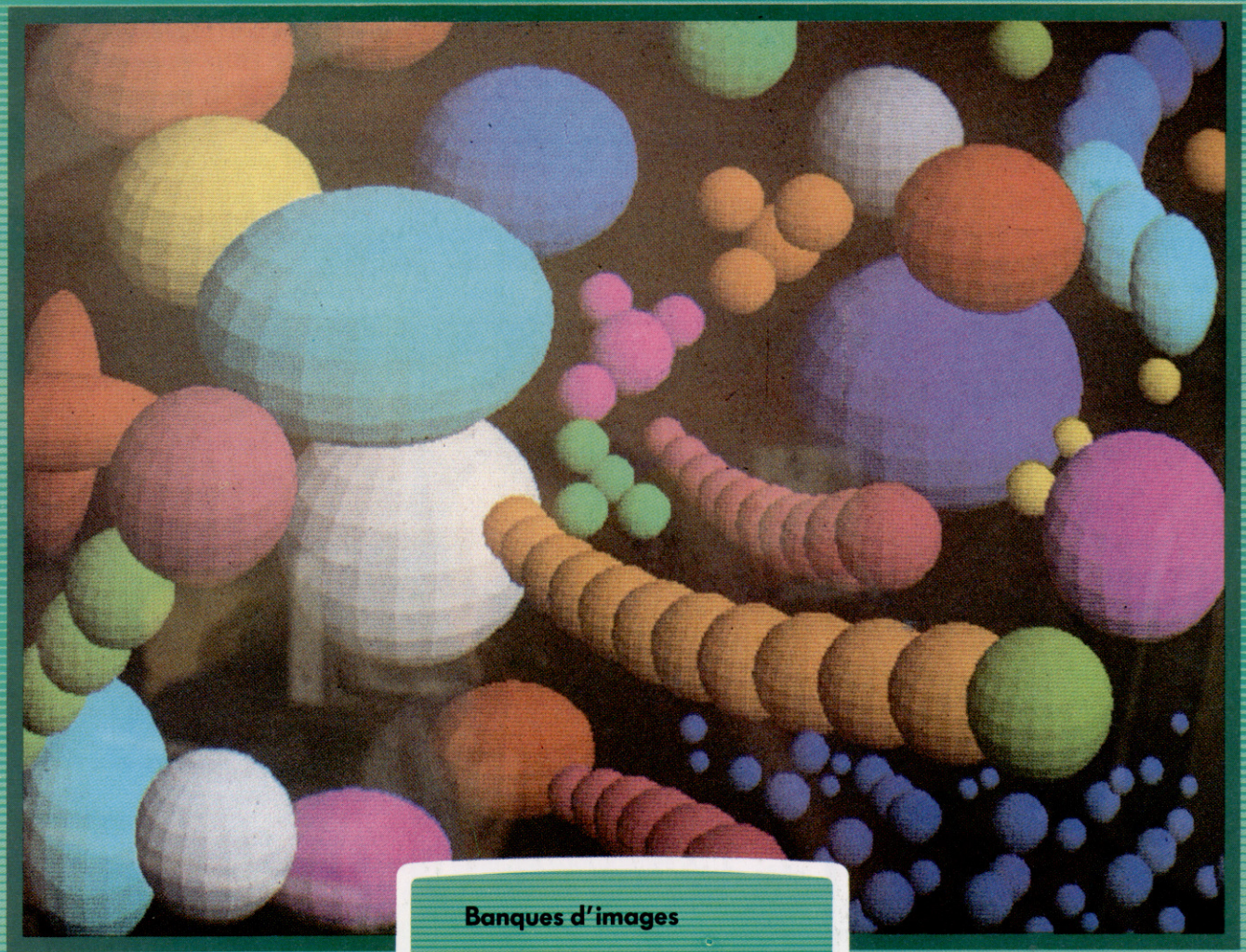
ABC INFORMATIQUE

N°35

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE



Banques d'images

Fichiers séquentiels

Le Sharp PC-5000

Le graphisme Commodore

EDITIONS ATLAS

M6062-35-12F 85FB-3,80FS-$1.95

aBc INFORMATIQUE

cours d'informatique pratique et familiale

EDITIONS ATLAS

Édité par ÉDITIONS ATLAS s.a., tour Maine-Montparnasse, 33, avenue du Maine, 75755 Paris *Cedex 15*. Tél. : (37) 35-40-23. Services administratifs et commerciaux : 3, rue de la Taye, 28110 Lucé. Tél. : (37) 35-40-23.

Belgique : ÉDITIONS ATLEN s.a., Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, Montréal Nord.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, Mezzovico.

Réalisé par EDENA s.a., 29, boulevard Edgar-Quinet, 75014 Paris.

Direction éditoriale : J.-Fr. Gautier. *Service technique et artistique* : F. Givone et J.-Cl. Bernar. *Iconographie* : J. Pierre. *Correction* : B. Noël.
Publicité : Anne Cayla. Tél. : 202-09-80.

VENTE AU NUMÉRO

Les numéros parus peuvent être obtenus chez les marchands de journaux ou, à défaut, chez les éditeurs, au prix en vigueur au moment de la commande. Ils resteront en principe disponibles pendant six mois après la parution du dernier fascicule de la série. (Pour toute commande par lettre, joindre à votre courrier le règlement, majoré de 10 % de frais de port.)

Pour la France, s'adresser aux services commerciaux des ÉDITIONS ATLAS. Tél. : (37) 35-40-23.

Pour les autres pays, s'adresser aux éditeurs indiqués ci-dessous.

SOUSCRIPTION

Les lecteurs désirant souscrire à l'ensemble de cet ouvrage peuvent s'adresser à :

France : DIFFUSION ATLAS, 3, rue de la Taye, 28110 Lucé. Tél. : (37) 35-40-23.

Belgique : ÉDITIONS ATLEN s.a., 55, avenue Huart-Hamoir, 1030 Bruxelles. Tél. : (02) 242-39-00. Banque Bruxelles-Lambert, compte n° 310-0018465-24 Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, 11450 boulevard Albert-Hudon, Montréal Nord, H 1G 3J9.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, zona industriale 6849 Mezzovico-Lugano. Tél. : (091) 95-27-44.

RELIEZ VOS FASCICULES

Des reliures mobiles permettant de relier 12 fascicules sont en vente chez votre marchand de journaux.

ATTENTION : ces reliures, présentées sans numérotation, sont valables indifféremment pour tous les volumes de votre collection. Vous les numéroterez vous-même à l'aide du décalque qui est fourni (avec les instructions nécessaires) dans chaque reliure.

En vente tous les vendredis. Volume III, n° 35.

ABC INFORMATIQUE est réalisé avec la collaboration de Trystan Mordrel *(secrétariat de rédaction)*, Jean-Pierre Bourcier *(coordination)*, Patrick Bazin, Jean-Paul Mourlon, Claire Rémy *(traduction)*, Ghislaine Goullier *(fabrication)*, Marie-Claire Jacquet *(iconographie)*, Patrick Boman *(correction)*.
Crédit photographique, couverture : C. Gayard-Réa.

Directeur de la publication : Paul Bernabeu. Imprimé en Italie par I.G.D.A., Officine Grafiche, Novara. Distribution en France : N.M.P.P. Tax. Dépôt légal : septembre 1984. 14849. Dépôt légal en Belgique : D/84/2783/27.

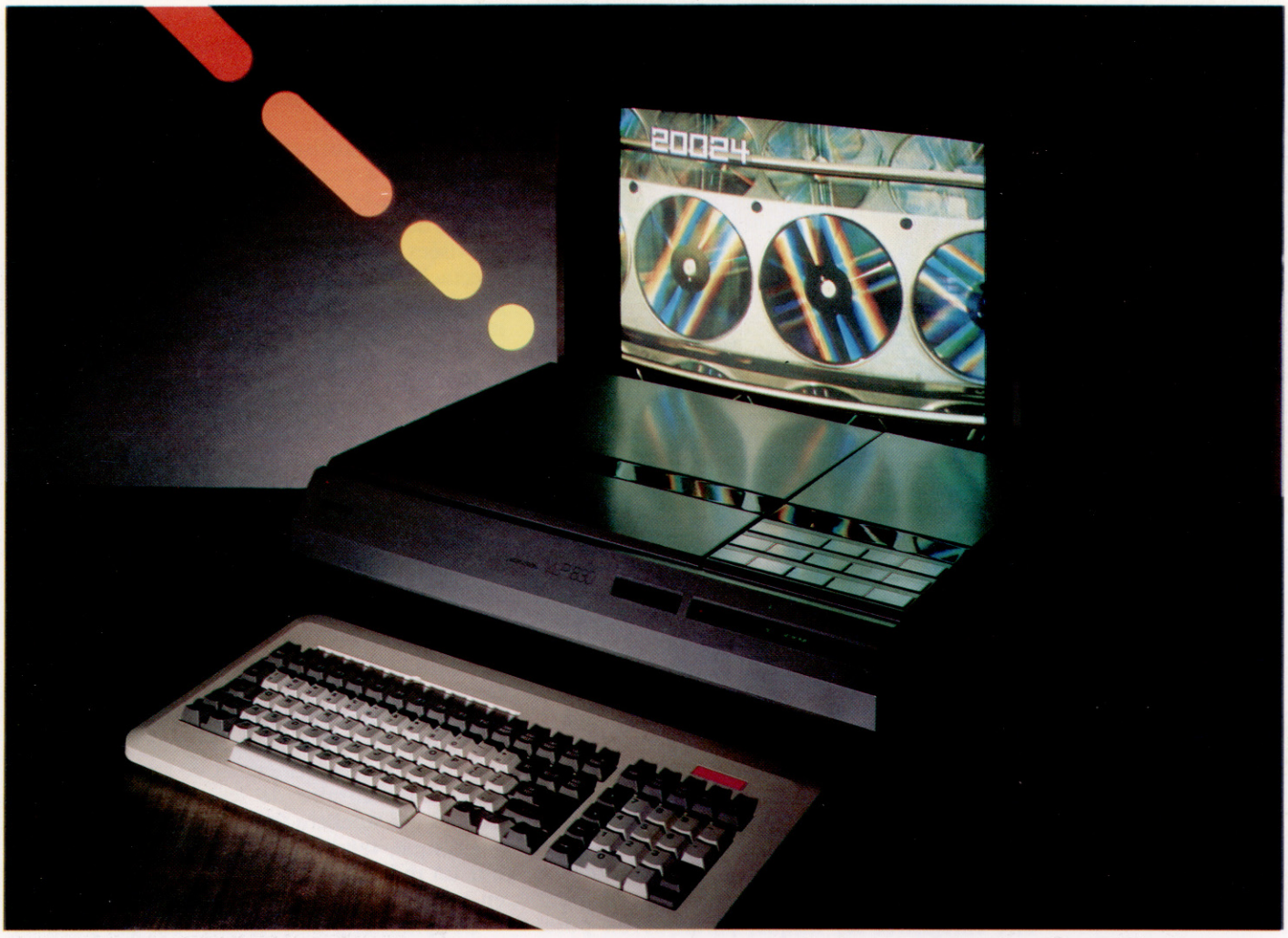
A NOS LECTEURS

En achetant chaque semaine votre fascicule chez le même marchand de journaux, vous serez certain d'être immédiatement servi, en nous facilitant la précision de la distribution. Nous vous en remercions d'avance.

Les Éditions Atlas

Banques d'images

Disquettes et disques vidéo sont des mémoires à haute densité et à accès direct. Grâce à une baisse de leurs prix, les mémoires trouvent des applications en micro-informatique.



Beaucoup de gens seront sans doute surpris d'apprendre qu'une platine à disques laser, qui était encore un article de luxe il y a quelques années, se vend désormais moins cher qu'un magnétoscope, tout en offrant une résolution bien meilleure. Une séquence télévisée est une image dynamique, qui peut être enregistrée sur une bande vidéo sous la forme d'une scène ininterrompue. Le seul moyen de retrouver une certaine image parmi d'autres consiste à rembobiner la bande, en se repérant à l'aide du compteur. Sur un disque laser, par contre, elle est stockée séparément, ce qui permet de retrouver l'élément qui vous intéresse très rapidement et sans risques d'erreur. En effet, l'emplacement de chaque image peut être défini en termes de piste et de secteur, et un microprocesseur peut parfaitement tenir à jour un catalogue très précis de chacun d'eux. Il lui est possible de fournir des images fixes ou projetées au ralenti, ainsi qu'un son stéréophonique.

Il doit aussi maintenir constante la vitesse de défilement de la platine, et placer la tête de lecture à certains endroits bien précis. Mais ce ne sont pas là des tâches très ardues, technologiquement parlant. Le plus difficile fut d'accroître, dans d'énormes proportions, la capacité mémoire des disques. Stocker des milliers d'images sur un objet de la taille d'un 33 tours est une entreprise pleine de difficultés.

Si votre ordinateur dispose de la haute résolution, vous n'ignorez pas que l'image de votre écran télé est composée de points individuels, les pixels. Plus on peut en placer sur l'écran,

L'âge du laser
Les platines à disques laser seront bientôt aussi accessibles que les micro-ordinateurs, et il sera courant de voir les deux reliés. L'achat de disques et de logiciels appropriés vous permettra d'accéder à des banques d'images, des programmes éducatifs sophistiqués, ou des jeux d'aventure animés. (Cl. Ian McKinnell.)

Nouvelle mémoire
Les disques à laser de 30 cm de diamètre et les disques compacts, plus petits, peuvent stocker en mémoire des informations visuelles ou sonores, ainsi que les données numériques enregistrées sur cassettes ou sur disquettes. Malheureusement il n'est pas possible d'« écrire » sur de tels disques qui ne peuvent qu'être lus, et vous serez contraints d'acheter des programmes pré-enregistrés sans pouvoir sauvegarder les vôtres. (Cl. Ian McKinnell.)

plus la résolution est élevée. Vous savez aussi que les images ainsi obtenues consomment énormément d'espace mémoire. C'est ainsi que pour une résolution maximale de 640 × 256, un écran complet peut occuper 20 K de mémoire... A supposer qu'une image télévisée soit affectée des mêmes caractéristiques, une seconde de vidéo (soit 25 images) exigerait 25 × 20 K, une minute 30 méga-octets (30 millions d'octets), et un épisode de feuilleton télévisé 1 giga-octet (1 000 méga-octets) de mémoire! Les micro-ordinateurs domestiques ont généralement recours à des disquettes simple densité : il en faudrait plus de six mille, et l'équivalent d'une semaine complète de travail...

Dans ces conditions, il semble plus avantageux de recourir à un stockage sur bande vidéo; une demi-heure d'un programme télévisé, pour être enregistrée sur disque, paraît devoir poser des problèmes insurmontables.

On ne peut y parvenir qu'en écrivant les données de façon qu'elles occupent le moins de place possible. Elles sont gravées, au moyen d'un rayon laser de l'épaisseur d'un cheveu, sur une surface métallique incluse dans une solide enveloppe protectrice en plastique transparent. Un autre rayon laser, à basse puissance, peut ensuite les lire. Dans les deux cas on se sert de la technologie laser, parce que c'est un procédé à haute résolution et à basse tolérance. Aucun autre procédé ne pourrait lire ou écrire des données dans un espace aussi restreint.

La mise en œuvre proprement dite est une combinaison des méthodes relatives à la haute-fidélité et aux lecteurs de disquettes. Les sillons d'un disque conventionnel forment une spirale continue, et les parois du sillon abritent sous forme gravée le message musical enregistré. Les pistes d'une disquette, en revanche, se présentent sous forme de cercles concentriques; l'information est inscrite magnétiquement, et stockée sous forme numérique (des suites de zéros et de uns). Le disque laser fait usage de pistes, et non de sillons, mais elles forment une spirale. Les données y sont gravées optiquement, mais ne peuvent être effacées. Elles aussi sont constituées de uns et de zéros : dans le premier cas, le rayon laser creuse une minuscule dépression sur la surface métallique, dans le second cas il la laisse intacte. Chaque creux n'a qu'un demi-micron de largeur et un dixième de micron de profondeur. La surface du disque peut donc en accueillir quatre cents millions par centimètre carré...

Cette miniaturisation, si étonnante qu'elle soit, suffit à peine à répondre aux exigences mémoire de type vidéo. Un disque de 35 centimètres de diamètre peut accueillir 54 000 images télévisées par face, soit environ 36 minutes de spectacle. Les calculs que nous avons effectués précédemment partaient d'une haute résolution moyenne, et en noir et blanc, alors qu'un disque laser doit conserver non seulement la couleur individuelle de chaque point de l'écran, mais aussi une piste sonore! Une seule vue en couleur, accompagnée de la piste sonore qui lui correspond, représente dans ces conditions près de 100 K d'espace mémoire, et 54 000 images consommeront donc 5,4 millions de K.

Ces énormes possibilités ouvrent aux disques laser de vastes champs d'application. Leur point fort est sans doute la suppression d'un des plus gros obstacles des traitements de données : la collecte et l'entrée des informations. Il faut d'habitude que quelqu'un s'assoie devant un terminal et les tape sous forme codée, afin qu'elles soient compréhensibles par le système. C'est là une méthode fastidieuse, onéreuse et qui consomme énormément de temps. S'il vous suffit au contraire de les placer devant une caméra qui stockera toute l'information visuelle sur disque, tandis que vous vous bornez à établir un index des images enregistrées, le travail en sera, bien sûr, grandement facilité.

Les platines à disques laser ont des possibilités très différentes. Les moins chères sont déjà sur le marché, et sont destinées au simple particulier. On peut s'en servir comme d'un magnétoscope pour passer des films, mais la qualité de l'image est infiniment supérieure; de plus, ce type d'appareil peut fonctionner en ralenti ou montrer une image particulière. Sur certains modèles il est même possible de sélectionner telle ou telle vue, grâce à un numéro d'ordre ou à des procédures de recherche. L'écran se vide un instant, puis l'image désirée apparaît.

Pour tirer pleinement parti de toutes ces potentialités, il faut pouvoir opérer un choix grâce à un programme d'ordinateur. Pioneer et Philips ont déjà mis en vente de telles platines, mais elles sont coûteuses et destinées seulement à un emploi professionnel. La méthode la plus simple consiste à utiliser une interface IEEE ou

RS232, afin que l'ordinateur puisse faire une sélection d'images grâce à un numéro de code; un logiciel approprié peut conserver dans une base de données la liste de toutes les vues disponibles, et y puiser selon les besoins.

Philips a repris cette idée, et ses platines les plus récentes abritent un microprocesseur simple. Il peut charger en mémoire un programme adapté à tel ou tel disque à partir d'une cartouche EPROM enfichée sur l'appareil ou même du disque lui-même. Ce dernier est pourvu d'une piste vidéo et de deux pistes audio : la piste sonore peut ainsi être en deux langues différentes : mais l'une d'elles, si elle reste inutilisée, peut servir à l'accueil d'un programme d'ordinateur.

Nous disposons donc d'une banque de 54 000 images de qualité, sous contrôle de l'ordinateur. Le stade final consiste à mélanger les images du disque et le texte tapé sur l'appareil. On peut pour cela se servir de deux moniteurs différents, additionner deux signaux vidéo, ou mieux encore faire usage d'un moniteur de type Minitel. On obtient ainsi quelque chose de tout à fait nouveau : la vidéo interactive. L'utilisateur peut lire le disque dans n'importe quel ordre, et donc commander l'affichage de vues ou de séquences selon son désir.

Il est évident que cela rend possible la création de banques d'images. On pose certaines questions à l'ordinateur, qui travaille ensuite sur une base de données et transmet à la platine l'ordre d'afficher l'image demandée.

La vidéo interactive peut aller encore plus loin : l'utilisateur peut jouer un rôle dans les séquences projetées sur l'écran. Un programme éducatif peut expliquer un point de détail à l'aide d'un bref extrait filmé, puis poser des questions, revenir en arrière, donner des précisions supplémentaires, et ainsi de suite. On pourra même produire des films que chacun dirigera en prenant les décisions qu'imposera la conduite des personnages. L'intrigue et la fin seront différentes suivant votre façon de jouer. De telles réalisations sont déjà en cours.

Bien entendu, il faudra résoudre bien des problèmes avant que ce marché puisse se développer. Mis à part les questions de prix de revient (des disques comme des platines), il est nécessaire de repenser complètement l'écriture des programmes, tout comme la mise au point des scénarios et le tournage des films. Bien des petites compagnies cherchent déjà à relever le défi et se placent sur ce nouveau créneau de la communication interactive. Elles proposent à leurs clients un éventail de services complet — choix et installation de l'équipement, création et production des disques, rédaction de logiciels. Elles entendent bien couvrir toutes les applications de la vidéo interactive. Mais les prix sont encore très élevés.

Les possesseurs de micro-ordinateurs ne doivent pourtant pas désespérer. Les disques à lecture laser sont encore coûteux, mais guère plus, tout compte fait, que les films et les magnétoscopes pris ensemble. Prenons des exemples : les disques compacts se vendent déjà aux alentours de 150 F; des disques interactifs pourraient, sans doute, être proposés entre 250 et 300 F, logiciels inclus. Ce sont là des prix assez raisonnables, vu les immenses possibilités offertes.

Vidéo interactive
De nombreuses platines à disques laser sont pourvues d'une interface IEEE ou RS232, qui permet de les contrôler à partir d'un ordinateur. Une configuration typique pourrait rassembler les deux types d'appareil connectés à une base de données stockées sur disquette et relatives aux images contenues sur le disque. L'utilisateur fait son choix dans ce catalogue, et l'ordinateur donne à la platine l'ordre de rechercher et d'afficher telle vue, grâce à un numéro d'ordre. L'illustration ci-dessous permet de voir comment combiner sur un seul écran les images du disque et le texte tapé sur le clavier. Un système plus simple aurait recours à deux écrans.
(Cl. Steve Cross.)

Fichiers en règle

L'organisation séquentielle — ou en série — des fichiers est un héritage du traitement informatique sur bandes magnétiques. Nous voyons dans cet article comment créer des fichiers séquentiels.

Un fichier binaire (ou programme) est une forme simplifiée de fichier séquentiel. Lorsque vous tapez SAVE«programme» (sauvegarde), le système d'exploitation assure, sans même que vous vous en rendiez compte, les diverses opérations nécessaires à la création d'un fichier. En premier lieu, la communication est établie avec l'unité de disques ou de cassettes. Le système d'exploitation écrit un label de bande qui indique le nom du fichier ainsi que certaines informations sur son contenu. Ensuite, le bloc mémoire comportant le programme courant est écrit au fichier. En dernier lieu, un marqueur de fin de fichier est attribué au fichier; il coupe la communication entre les lecteurs (de disques ou de cassettes) et l'ordinateur.

Les commandes BASIC servant à créer un fichier séquentiel diffèrent d'un ordinateur à l'autre, même si elles doivent effectuer des tâches identiques. Prenons comme exemple le Commodore 64 :

```
100 R$ =«voici un enregistrement du fichier»
150 OPEN 5,1,2, «FICHIER DE DONNÉES, SEQ, ÉCRITURE»
200 PRINT#5,R$
250 CLOSE 5
```

Cette séquence commence par la commande OPEN 5,1,2, qui indique au système d'exploitation d'ouvrir un canal de communication, appelé canal 5, vers le périphérique numéro 1 — le lecteur de cassettes. Les fichiers séquentiels peuvent être créés sur divers périphériques et l'on peut accéder à plusieurs fichiers en même temps. Aussi le canal allant de l'ordinateur au périphérique et, par là même, vers un fichier donné, doit être identifiable par un numéro. Le nombre 2 est propre au système Commodore et ne concerne pas notre exposé.

Après la commande OPEN, on indique le nom du fichier. Sur le Commodore, il s'agit d'un nom comme «FICHIER DE DONNÉES», suivi du type du fichier (SEQ pour séquentiel), et du type d'accès (ÉCRITURE indique que le fichier est ouvert — OPEN — pour écriture). Les fichiers séquentiels peuvent être ouverts soit pour écriture, soit pour lecture, mais pas pour les deux à la fois. L'effet de cette commande est d'activer la tête de lecture-écriture de l'unité concernée, et d'inscrire un enregistrement d'en-tête (nom du fichier) avec certaines informations en début de fichier. Cela permet de signaler le commencement du fichier.

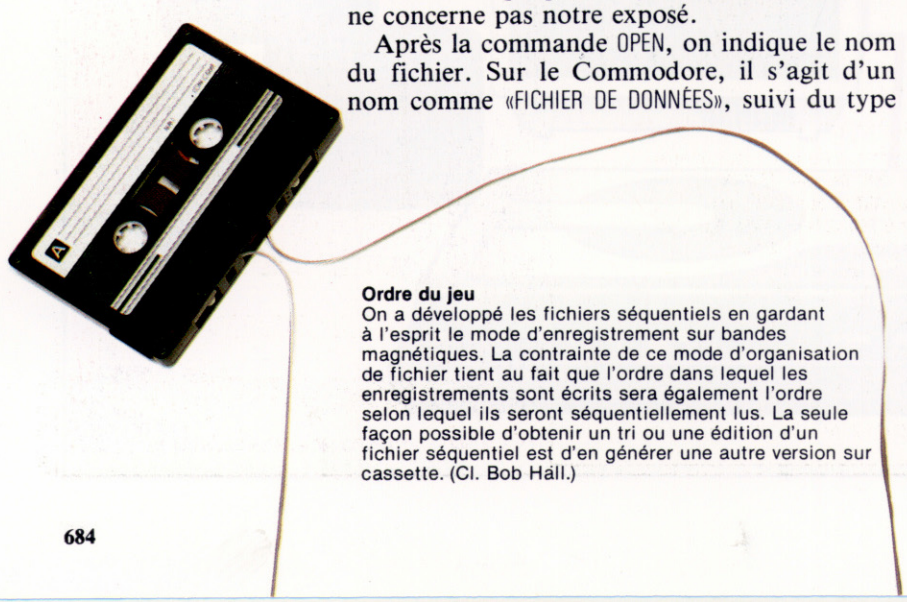
La commande PRINT#5 à la ligne 200 envoie des données au fichier. PRINT a sa signification habituelle, et # indique que les données doivent être envoyées sur le canal spécifié plutôt qu'à l'écran (le canal de sortie par défaut). Le contenu de R$ est donc envoyé sur le canal 5 vers le fichier séquentiel appelé «FICHIER DE DONNÉES». Le fichier comporte alors un enregistrement : voici un enregistrement du fichier. Enfin, CLOSE 5 ferme le canal 5 et dispose un marqueur « fin de fichier » après le dernier enregistrement.

L'information que contient le fichier est destinée à être lue ultérieurement, par exemple :

```
500 OPEN 3,1,2 «FICHIER DE DONNÉES,SEQ,LECTURE»
550 INPUT #3,A$
600 CLOSE 3
650 PRINT A$
```

Ce segment de programme utilise comme le précédent la commande OPEN; elle signifie ici « trouver le début du fichier — appelé FICHIER DE DONNÉES — et préparer la lecture ». Dans le programme précédent, OPEN signifiait « créez un fichier sous le nom de FICHIER DE DONNÉES et préparez l'écriture sur ce fichier ». Les numéros de canaux sont différents, mais il s'agit simplement d'étiquettes (on peut prendre pour chaque opération un numéro de canal différent). L'adresse du périphérique est bien entendu la même, ce qui permet de retrouver le fichier (unité de lecteur de cassette 1). Le nom du fichier et son type doivent également être les mêmes (ils l'identifient). Par contre, le type d'accès est différent, LECTURE au lieu d'ÉCRITURE.

Ce programme comporte INPUT#3,A$, ce qui signifie que l'entrée des données se fait *via* le canal 3, et non depuis le clavier (canal d'entrée par défaut). Le premier enregistrement complet du fichier est lu et envoyé par le canal 3 sur la variable A$, de la même manière qu'il était envoyé par le canal 5 depuis la variable A$, vers le fichier, par PRINT#5,R$, au segment de programme précédent.



Ordre du jeu
On a développé les fichiers séquentiels en gardant à l'esprit le mode d'enregistrement sur bandes magnétiques. La contrainte de ce mode d'organisation de fichier tient au fait que l'ordre dans lequel les enregistrements sont écrits sera également l'ordre selon lequel ils seront séquentiellement lus. La seule façon possible d'obtenir un tri ou une édition d'un fichier séquentiel est d'en générer une autre version sur cassette. (Cl. Bob Hall.)

Classer et trouver

```
199  REM++++++++++++ECRITURE FICHIERS++++++++++++
200  REM+         ECRIRE       FICHIERS          +
201  REM++++++++++++CBM  C64++++++++++++++++++++++
220  GOSUB 1500
240  FOR K=65 TO 90
260  Z$=CHR$(K) +X$:C$=D$+"WRITING "+CHR$(K)
280  GOSUB 2000
300  NEXT K
399  REM++++++++++++++++++++++++++++++++++++++++
400  REM+          LECTURE     FICHIERS          +
401  REM++++++++++++++++++++++++++++++++++++++++
420  FOR L=0 TO 1 STEP 0:FOR M=1 TO 1
440  PRINT D$; "ACCES AUX ENREGISTREMENTS"
460  INPUT"RECHERCHE CHAINE (*=QUIT ) ";N$
480  L$=LEFT$(N$,1) : IF L$="*" THEN GOTO 5000
500  IF L$<"A" OR L$>"Z" THEN M=0
520  NEXT M
540  Z$=L$+Y$
560  GOSUB 3000
580  PRINT TAB(5) "RECORD";N; "(FRAPPEZ UNE TOUCHE)"
600  GET GT$:IF GT$="" THEN 600
620  NEXT L
999  END
1499 REM*****************************************
1500 REM*********INITIALISER S/R*****************
1501 REM*****************************************
1520 D$=CHR$(147):PRINT D$,CHR$(8);CHR$(142)
1540 X$=",S,W":Y$=",S,R"
1600 RETURN
1999 REM*****************************************
2000 REM*********ÉCRITURE  FICHIER S/R***********
2001 REM*****************************************
2020 PRINT C$:INPUT"COMBIEN D'ENREGISTREMENTS";R
2040 OPEN 8,8,2,Z$
2060 IF R=0 THEN PRINT#8,"*":CLOSE8:RETURN
2080 FOR I=1 TO R
2100 PRINT C$:PRINT "ENREGISTREMENT #";
2120 INPUT "TEXT.....";R$
2140 PRINT#8,R$
2160 NEXT I
2180 PRINT#8,"*":FERMETURES
2499 RETURN
2999 REM*****************************************
3000 REM**********LECTURE  FICHIER S/R***********
3001 REM*****************************************
3020 PRINT D$;"RECHERCHE";L$;" FOR ";N$
3040 OPEN 8,8,2,Z$
3060 FOR I=1 TO 100000
3080 INPUT#8,R$
3100 PRINT R$
3120 IF R$="*" THEN N=0:I=100000
3140 IF R$=N$ THEN N=I:I=100000
3160 NEXT I:FERMETURE8:N$=""
3180 RETURN
5000 REM******FERMETURE  PROGRAMME******
5020 PRINT CHR$(9) "FIN DU PROGRAMME":STOP
```

220 : Initialise.

260-280 : Crée les fichiers «A» à «Z».

420-540 : Entrée de l'enregistrement à rechercher, trouver la première lettre du fichier et identifier le fichier.

560 : Recherche de ce fichier.

580-600 : Transmettre le résultat des recherches.

1520 : Effacer l'écran, passer en mode majuscules.

2040 : Ouverture du fichier pour écriture.

2060 : Vérifier que le fichier est vide, écrire «*», (fermeture du fichier).

2120-2140 : Entrer le texte de l'enregistrement et l'écrire au fichier.

2180 : Écrire en tant que dernier enregistrement «*», et fermer le fichier (CLOSE).

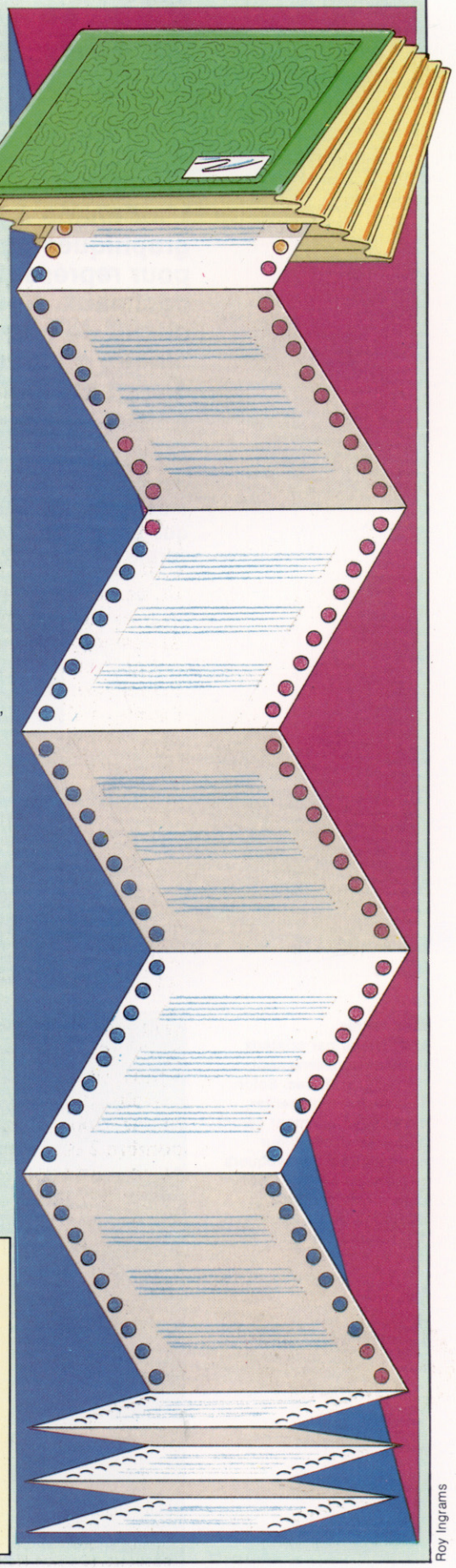
3040 : Ouverture du fichier en lecture (OPEN). pour le mode (READ).

3120 : Test « dernier enregistrement ? », arrêter les recherches.

3140 : Test « enregistrement trouvé ? », arrêter les recherches.

3160 : Fermeture du fichier (CLOSE).

Ce programme illustre l'utilisation de fichiers séquentiels sur disque par la création d'une simple table d'index comprenant 26 fichiers, c'est-à-dire un par lettre de l'alphabet. Chaque fichier comporte des enregistrements dont la première lettre correspond à la lettre de l'alphabet. Vous pouvez ou non créer des enregistrements. Lors de vos recherches, le fichier concerné sera passé en revue jusqu'à ce que vous trouviez l'enregistrement en question. S'il n'est pas trouvé, le message « ENREGISTREMENT 0 » sera affiché.
Le programme utilise le SED afin d'accéder directement au fichier.



Roy Ingrams

Variantes de basic

BBC Micro

Reprenez les variantes des lignes 260 et 540 pour le Spectrum. Remplacez PRINT#8, INPUT#8 et CLOSE8 par PRINT#C8, INPUT#C8 et CLOSE#C8.

```
600 GT$=GET$
1520 *DISK
1530 MODE 7
1540 D$=CHR$(12):PRINT D$«UTILISEZ LES MAJUSCULES»
1550 PRINT«-FRAPPEZ UNE TOUCHE-»:GT$=GET$
2040 C8=OPENOUT(Z$)
3040 C8=OPENIN(Z$)
5020 PRINT «FIN DU PROGRAMME»
```

Spectrum micro-lecteur

Insérez LET partout où cela est nécessaire.
Remplacez PRINT D$;.. par CLS PRINT..
Remplacez PRINT C$ par CLS:PRINT:C$
Remplacez OPEN 8,8,2,Z$ par OPEN #8;«m»;1;Z$
Remplacez CLOSE8 par CLOSE #8
Supprimez la ligne 1540

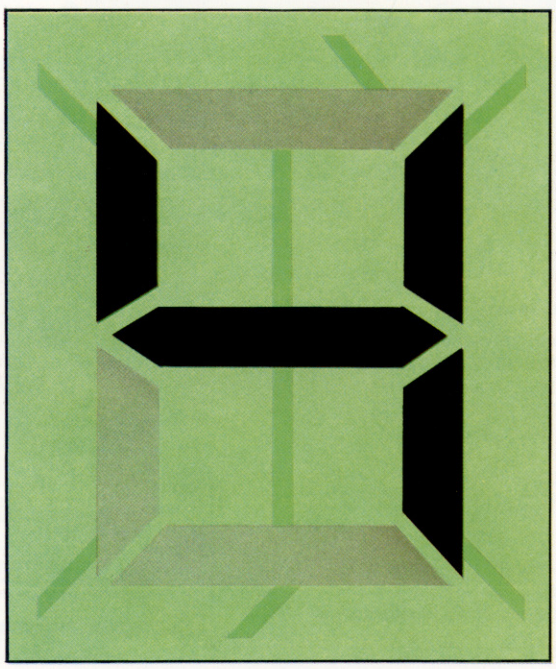
```
260 Z$=CHR$(K):LET C$=«ÉCRITURE»+Z$
480 LET L$=N$(1):IF L$=«*» THEN GOTO 5000
540 LET Z$=L$
600 PAUSE 0
1520 CLS:LET F2=PEEK 23658:POKE 23658,8
3080 INPUT #8;R$
5020 POKE 23658,F2:PRINT«FIN DU PROGRAMME»:STOP
```

Des chiffres s'allument

L'affichage à sept segments graphiques par chiffre est utilisé pour représenter des nombres décimaux. Nous étudions un circuit de conversion d'un code binaire en un code d'affichage à sept segments.

Nous supposerons que la représentation binaire des nombres décimaux se fait selon le code dit « décimal codé binaire », ou DCB. Cela signifie que chaque chiffre décimal est stocké sous la forme de quatre chiffres binaires :

Décimal codé binaire	Décimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010 à 1111	inutilisé



Sept barres
L'affichage par cristaux liquides concerne sept barres graphiques. Selon le chiffre à représenter, certaines seulement seront allumées et c'est leur combinaison qui évoquera le tracé du chiffre. Chaque barre porte un nom distinctif, de *t* à *z*. (Cl. Kevin Jones.)

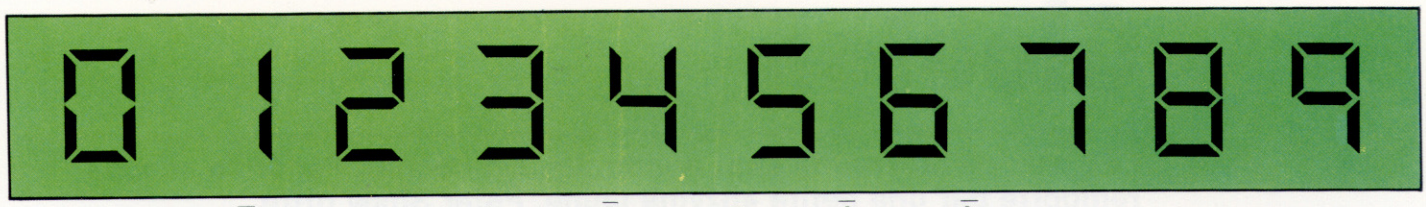
Le mode d'affichage dit « à sept segments » sert à représenter des chiffres décimaux en faisant s'allumer certaines diodes électroluminescentes (LED), ou en changeant la polarité de certaines barres dans le cas d'un affichage par cristaux liquides (LCD). L'illustration montre comment s'appellent les segments (la barre supérieure s'appelle *t*, l'inférieure, *w* etc.).

Nous disposons ainsi de toute l'information nécessaire pour établir une table de vérité. La figure qui représente le chiffre 4 montre quelles sont les barres à activer lorsque le circuit reçoit chaque entrée binaire. Lorsque cette dernière est de 0100, par exemple (4 en valeur décimale), le circuit allume ou active les barres *u*, *v*, *y* et *z*. De la même manière, 1000 (8 en décimal) fera s'allumer et donc apparaître toutes les barres. La table de vérité pour le circuit sera la suivante :

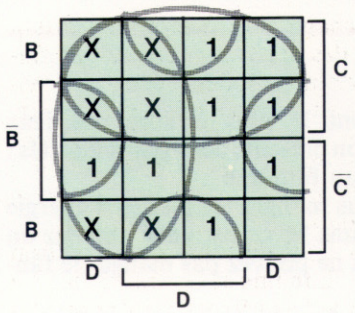
Chiffres décimaux	Entrées				Sorties						
	A	B	C	D	t	u	v	w	x	y	z
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
	1	0	1	0	X	X	X	X	X	X	X
	1	0	1	1	X	X	X	X	X	X	X
	1	1	0	0	X	X	X	X	X	X	X
	1	1	0	1	X	X	X	X	X	X	X
	1	1	1	0	X	X	X	X	X	X	X
	1	1	1	1	X	X	X	X	X	X	X

Il n'y a malheureusement aucun moyen simple pour analyser cette table et l'on devra étudier chaque sortie séparément pour la simplifier. Nous établirons donc une table de Karnaugh pour chaque sortie, les conditions non pertinentes (X) — «affichage sans intérêt» — devant figurer pour chaque table. Dans certains cas, cela contribuera à une simplification. Les sept tables de Karnaugh sont données plus loin, une par barre.

La première table de Karnaugh, par exemple, traite des entrées qui activeront la barre *t* (cette barre est utilisée pour tous les chiffres sauf deux). Nous serons alors à même d'établir le circuit à partir des expressions simplifiées des lignes de sortie. Ce circuit ne génère qu'une

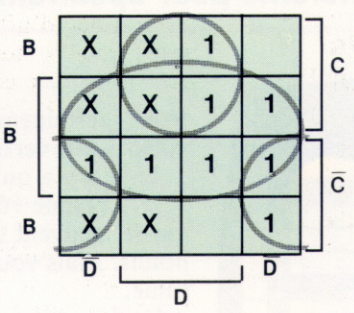


Pour t



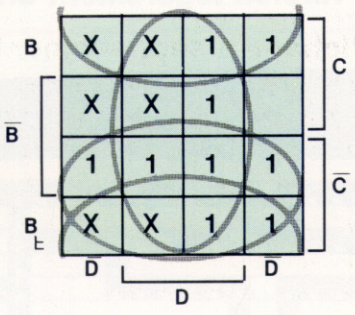
$t = C + A + \bar{B}.\bar{D} + B.D$

Pour u



$u = \bar{B} + C.D + \bar{C}.\bar{D}$

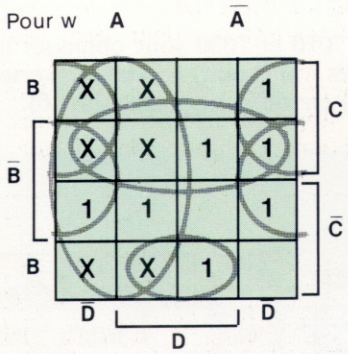
Pour v



$v = D + \bar{C} + B$

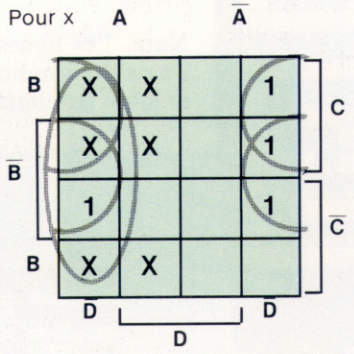
Codage des barres
Les diagrammes suivants permettent de dire quelles barres doivent être activées pour obtenir des chiffres. Le chiffre 1 s'obtiendra en allumant les barres *u* et *v*. La table de vérité de la page précédente indique les lignes de sortie pour chaque chiffre.

Pour w



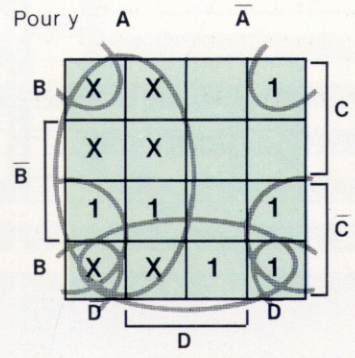
$w = A + C.\bar{D} + \bar{B}.C + \bar{B}.\bar{D} + B.\bar{C}.D$

Pour x



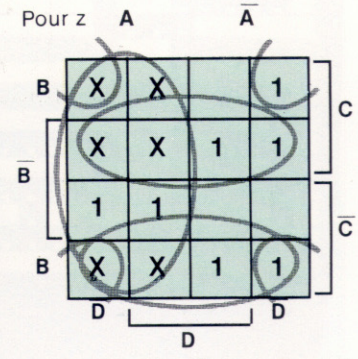
$x = C.\bar{D} + A.\bar{D} + \bar{B}.\bar{D}$
$= (A + \bar{B} + C).\ \bar{D}$

Pour y

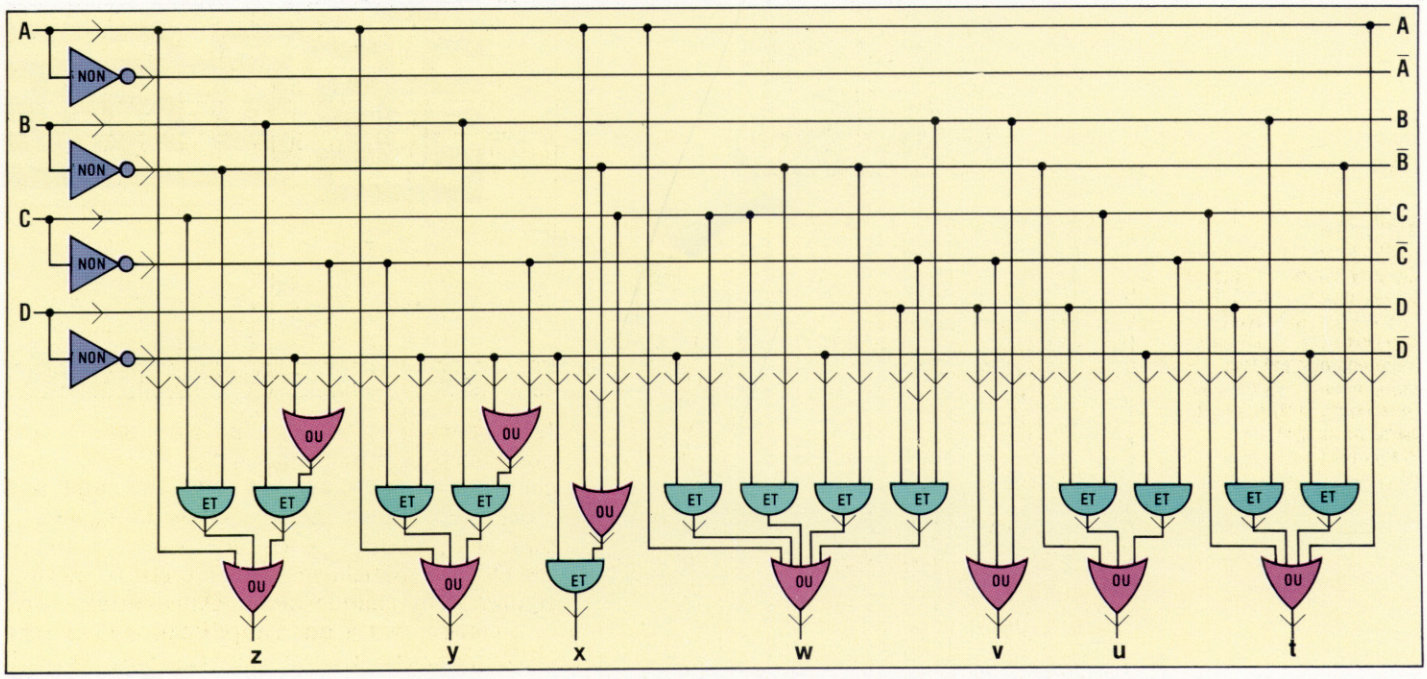


$y = A + B.\bar{C} + \bar{B}.\bar{D} + \bar{C}.\bar{D}.$
$= A + B.(\bar{C} + \bar{D}) + \bar{C}.\bar{D}$

Pour z



$z = A + \bar{B}.C + B.\bar{C} + B.\bar{D}$
$= A + B.C + B.(\bar{C} + \bar{D})$

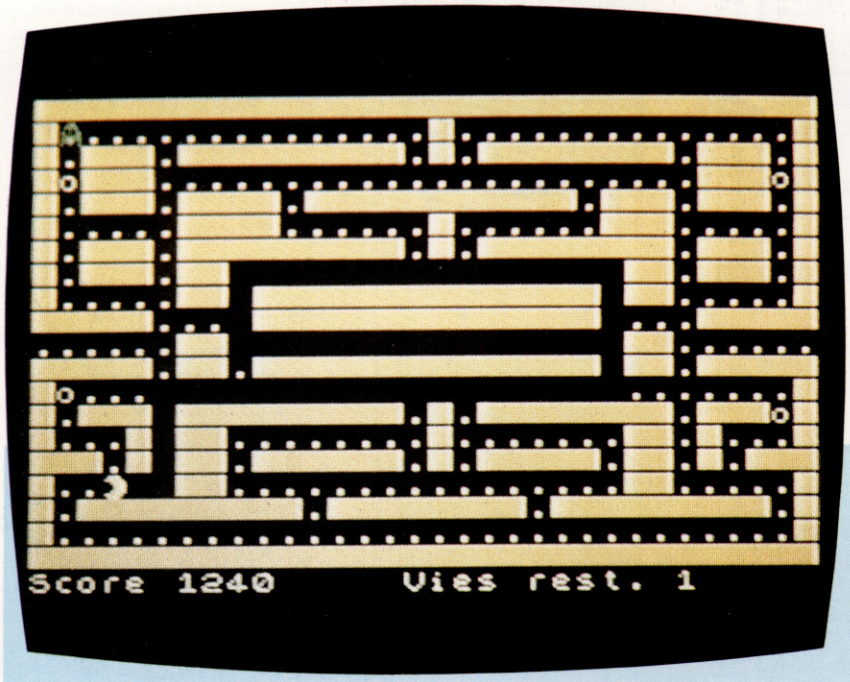


Liz Dixon

seule position d'affichage à la fois. Du fait que la plupart des circuits de ce genre concernent huit ou même dix positions d'affichage à la suite, il semble qu'il faudrait répéter autant de fois ce circuit qu'il y a de positions d'affichage. Il est cependant possible de mettre en commun le convertisseur : cela s'appelle le multiplexage. Ainsi, il n'y a qu'une seule position à la fois qui reçoit des informations du circuit (configuration des barres), l'affichage de l'ensemble des positions étant néanmoins perçu grâce à la vitesse très élevée de la commutation.

Glouton

Les jeux de labyrinthe sont nombreux. On connaît l'énorme succès remporté par une petite enzyme jaune. Peter Shaw offre une version légèrement différente pour Spectrum.



Voici certainement le plus célèbre des jeux vidéo. Cette version présente deux particularités :

1. Il n'y a qu'un fantôme.
2. Lorsque vous mangez les pilules d'énergie qui se trouvent dans les coins, vous obtenez un bonus. Mais vous ne pouvez pas manger le fantôme.

Utilisez les touches de déplacement du curseur pour vous déplacer.

Note. Les lignes 7010 et 7020 sont obtenues à l'aide des symboles «.» (point) en vidéo inverse et «—» (souligné) en vidéo normale (shift 0).

```
  20 LET rec=0: GO SUB 9000
  30 GO SUB 8000
  40 LET w=12: LET h=16: GO SUB
1000: PRINT AT 20,15; INVERSE 1;
Vies rest. ";vies
  45 PRINT AT w,h; INVERSE 1;m$
  50 LET a$=INKEY$: IF a$<"5" OR
a$>"8" THEN GO TO 80
  60 IF a$<>"" THEN LET c1=VAL a
$

  70 LET m$=("■" AND c1=7)+("■"
AND c1=8)+("■" AND c1=6)+("■" AN
D c1=5)
  80 PRINT AT w,h;"■"
  85 IF h<1 THEN LET h=30
  86 IF h>30 THEN LET h=0
  90 LET w1=w: LET h1=h
 100 LET w=w+(c1=6)-(c1=7)
 110 LET h=h+(c1=8)-(c1=5)
 120 IF SCREEN$ (w,h)="_" THEN L
ET w=w1: LET h=h1: PRINT AT w,h;
INVERSE 1;m$: GO TO 150
 130 IF SCREEN$ (w,h)="." THEN L
ET sc=sc+10: BEEP .008,10: LET c
ompte=compte+1
 135 IF SCREEN$ (w,h)="o" THEN G
O SUB 6000
 140 IF SCREEN$ (w,h)="" THEN GO
TO 5000
 145 PRINT AT w,h; INVERSE 1;"■"
AT 20,0;"Score ";sc
 150 PRINT AT z,x;"■"
 160 LET z1=z: LET x1=x
 170 LET z=z+(m1=6)-(m1=7): LET
=x+(m1=8)-(m1=5)
 180 IF SCREEN$ (z,x)="_" THEN L
ET m1=INT (RND*4)+5: LET z=z1: L
ET x=x1: GO TO 160
 185 PRINT AT w,h; INVERSE 1;m$
 190 IF SCREEN$ (z,x)="." THEN P
RINT AT z1,x1;"■"
 195 IF SCREEN$ (z,x)="o" THEN P
RINT AT z1,x1;"■"
 200 IF SCREEN$ (z,x)="" THEN GO
TO 5000
 210 PRINT AT z,x; INVERSE 1; PA
PER 4;"■"
 230 IF vies<1 THEN GO TO 2000
 240 IF compte=tot THEN LET comp
te=0: GO TO 40
 250 GO TO 50
 990 STOP
2000 CLS
2010 PRINT '''TAB 5;"TERMINE"
2020 PRINT '''"          Votre score
:";sc
2030 IF sc>rec THEN LET rec=sc
2040 PRINT '''"    Record ";rec
2050 INPUT "Tapez ENTER pour rej
ouer"; LINE a$: GO TO 30
5000 INVERSE 1
5005 PRINT AT z,x;"."
5010 PRINT AT w,h;"■"
5020 BEEP .3,15
5030 PRINT AT w,h;"■"
5040 BEEP .3,11
5050 PRINT AT w,h;"■"
5060 BEEP .3,7
5070 PRINT AT w,h;"."
5080 BEEP .3,3
5090 PRINT AT w,h;":"
5100 BEEP .3,0
5110 PRINT AT w,h;" "
5120 BEEP .5,-5
5130 LET compte=0: LET v=12: LET
 h=16: LET vies=vies-1
5140 INVERSE 0: GO TO 40
6000 BEEP .008,15: BEEP .005,-15
: LET sc=sc+INT (RND*100)+200
6010 RETURN
7000 PAPER 0: CLS : INK 0: PAPER
 6: BORDER 0
```

```
7010 PRINT "
"
7020 PRINT "
"
7030 INK 0
7040 PAPER 6
7100 RETURN
8000 LET m$="■"
8020 LET c1=8: LET m1=8
8030 LET sc=0
8040 LET z=7: LET x=15
8050 LET vies=3
8060 LET compte=0
8070 LET tot=286
8990 RETURN
9000 FOR a=USR "a" TO USR "i"+7
9010 READ user: POKE a,user
9020 NEXT a: RETURN
9030 DATA 60,126,255,255,255,255
,126,60
9040 DATA 0,66,195,231,255,255,1
26,60
9050 DATA 60,126,248,240,240,248
,126,60
9060 DATA 60,126,255,255,231,195
,66,0
9070 DATA 60,126,31,15,15,31,126
,60
9080 DATA 0,0,126,255,255,255,12
6,60
9090 DATA 0,0,0,0,16,24,60,60
9100 DATA 129,66,36,0,0,36,66,12
9
9110 DATA 28,62,42,107,127,127,1
09,73
9980 REM a b c d e f g h i
9990 REM ■ ■ ■ ■ ■ ■ . ■ ■
```

Réponses

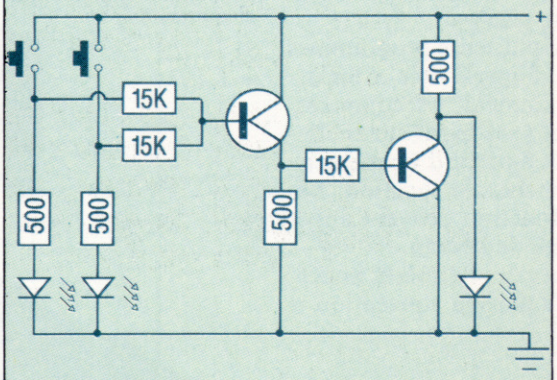
Nous vous avons proposé des exercices vous permettant de récapituler tout ce que nous avions abordé jusque-là. En voici les réponses. Elles peuvent ne pas correspondre exactement aux vôtres, car il y a de nombreuses façons de construire un circuit destiné à une tâche précise.

1. Résistances

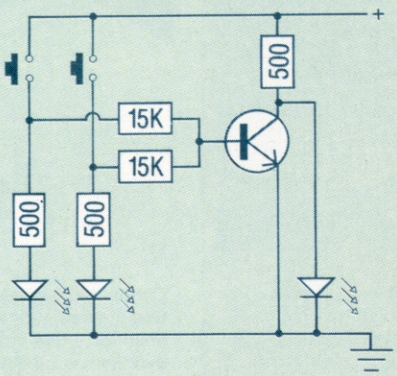
Les valeurs étaient respectivement de 6 400 KΩ(a) et de 150 K-Ω(b). Une résistance de 150Ω a des bandes de couleur successivement brune, verte et brune.

2. Porte NON-OU

La méthode la plus évidente pour construire une porte NON-OU consiste à combiner deux circuits OU et NON de la façon suivante :

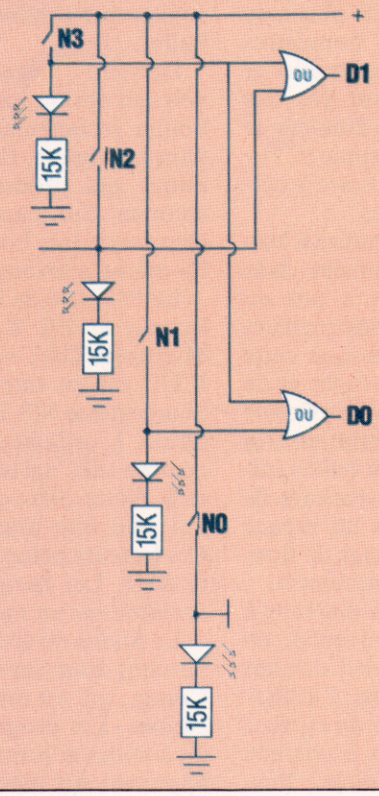


Mais il existe cependant un moyen plus rapide. On a recours à une porte OU et l'on extrait du collecteur du transistor (et non plus de l'émetteur) le signal de sortie, de façon similaire à une porte NON :



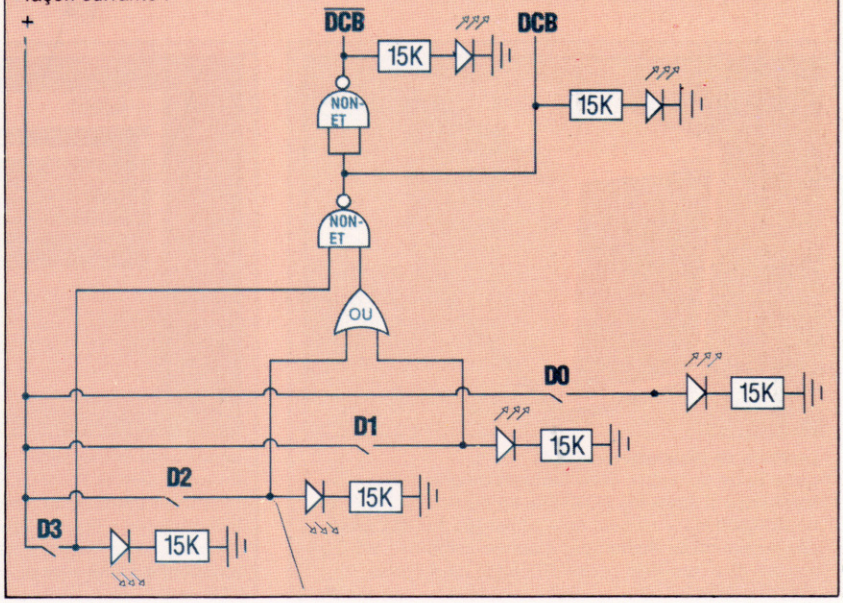
3. Convertisseur décimal/binaire

Un seul circuit intégré suffit : une puce TTL avec quatre portes OU.



4. Validité de nombres en DCB

Un nombre exprimé en DCB est valable s'il est compris entre 0000 et 1001, et illégal s'il est compris entre 1010 et 1111. Nous avons construit le circuit nécessaire de la façon suivante :



Liz Dixon

Les bulles arrivent

Sous le couvercle du PC-5000, machine MS-DOS portable créée par Sharp se cachent, entre autres caractéristiques, un affichage à cristaux liquides et une mémoire à bulles.

Les ordinateurs portables prennent plusieurs formes, mais les machines les plus populaires comportent un écran léger à cristaux liquides et peuvent être alimentées efficacement par des piles. Le PC-5000 de Sharp est l'ordinateur le plus coûteux de cette catégorie. Cet appareil utilise en plus les possibilités d'une mémoire à bulles au lieu d'un stockage sur cassette.

Le concept révolutionnaire de mémoire à bulles fut introduit il y a quelques années dans l'industrie informatique, mais il n'a pas eu l'impact attendu parce que peu de constructeurs ont réussi à résoudre les problèmes que cette technologie posait au niveau de la vitesse et de la fiabilité. Sharp semble avoir trouvé la bonne voie puisque sa machine test s'est montrée très rapide et très fiable. Une mémoire à bulles utilise très peu de courant et permet le stockage de grandes quantités de données dans un espace réduit. Le principe implique le stockage de données dans des bulles codées magnétiquement.

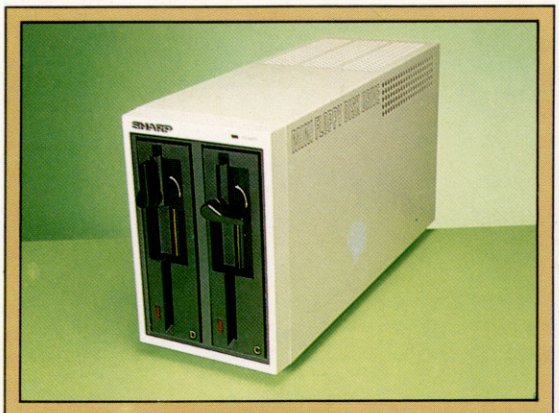
Le PC-5000 ressemble à une petite machine à écrire portable. En soulevant le couvercle, on découvre un clavier de machine à écrire incluant huit touches de fonction définies par l'utilisateur et quatre touches de commande du curseur placées en haut du clavier. L'écran à cristaux liquides se trouve dans le couvercle. Bien qu'assez lourd, celui-ci peut être orienté de diverses manières afin d'obtenir le meilleur confort d'utilisation. Trois voyants d'avertissement sont situés au-dessus du clavier (alimentation, épuisement des piles, bulles), ainsi qu'un connecteur destiné à un module de mémoire à bulles.

Derrière ce panneau, un support sert à recevoir, en option, une imprimante. Celle-ci est un boîtier rectangulaire qui s'adapte parfaitement dans le support. Il s'agit d'une imprimante thermique qui produit trente-sept caractères par seconde. Sur papier thermosensible, l'impression est de très bonne qualité bien que le papier lui-même n'arrange pas le résultat.

Le PC-5000 de Sharp a 128 K de mémoire utilisateur et 192 K de ROM, incluant le BASIC Microsoft GW, version utilisée sur l'IBM PC. L'UC est un Intel 8088, comme sur le PC, et Sharp utilise MS-DOS comme système d'exploitation. L'ordinateur adresse la mémoire à bulles comme s'il s'agissait de lecteurs de disquettes A et B. Sharp offre un lecteur de disquettes double qui peut être branché à l'arrière du PC-5000. Ces lecteurs peuvent être adressés avec les lettres C et D. Les lecteurs de disquettes ne peuvent fonctionner au moyen de piles.

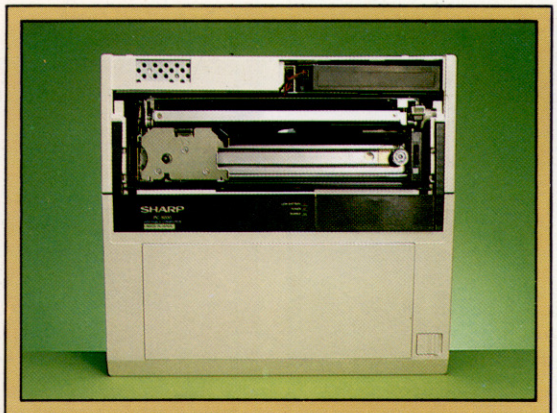
Le Sharp est livré avec plusieurs programmes Sorcim, dont SuperCalc, SuperWriter et SuperComm, qui est un programme de télécommunications. Les programmes sont livrés sur modules à bulles et sont appelés à partir d'un menu en appuyant sur une des touches de fonction. Les valeurs des touches de fonction peuvent apparaître sur la dernière ligne de l'écran.

Sharp a acquis au cours des dernières années une réputation de qualité dans la conception et la réalisation technique.



Ian McKinnell

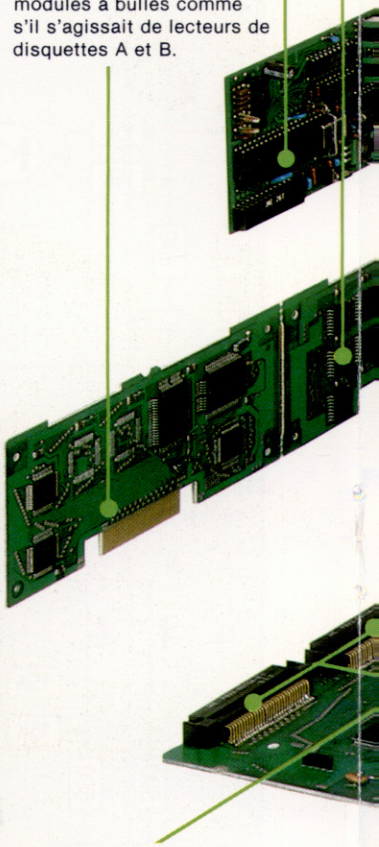
Lecteurs de disquettes doubles
Cette unité comporte deux lecteurs de disquettes fonctionnant sous MS-DOS. Elle se branche à l'arrière du PC-5000 mais ne peut être alimentée par des piles. Les programmes écrits pour d'autres machines MS-DOS doivent être formatés pour l'affichage de 8 lignes.



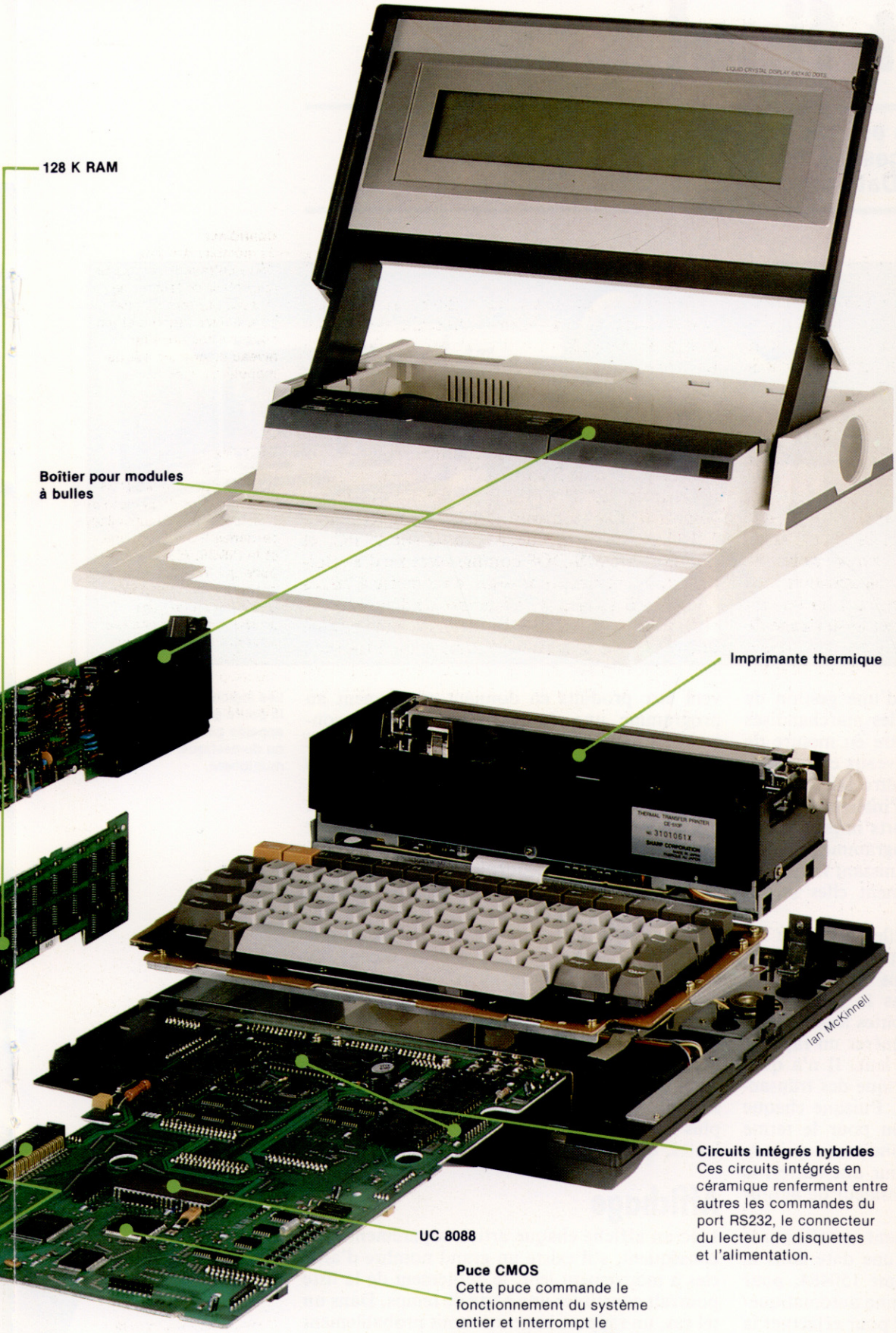
Imprimante optionnelle
En examinant le PC-5000, vous pouvez voir l'imprimante thermique optionnelle insérée dans le boîtier. L'imprimante s'insère dans le support et est connectée au système au moyen d'un câble-ruban. L'imprimante fonctionne à une vitesse de 37 caractères par seconde.

Contrôleur de mémoire à bulles
Cette unité agit comme un contrôleur de lecteur de disquettes, sauf qu'elle contrôle les entrées et les sorties effectuées au niveau du connecteur de module à bulles.

Carte ROM
Cette carte a 128 K de ROM, incluant le MS-DOS, un générateur de caractères d'affichage et d'impression, certaines E/S de système, et la PISCS. PISCS est une puce qui traduit les signaux provenant du module à bulles en un langage pouvant être compris par MS-DOS. Grâce à cette puce, MS-DOS lit les modules à bulles comme s'il s'agissait de lecteurs de disquettes A et B.



Connecteurs d'extension
Ces connecteurs sont destinés à des modules d'extension de RAM, ou à des programmes contenus dans des cartouches ROM.



SHARP PC-5000

PRIX

**

DIMENSIONS

300 × 318 × 90 mm.

POIDS

5,7 kg avec l'imprimante.

UC

Intel 16 bits 8088 et CMOS 8 bits.

MÉMOIRE

128 K de RAM.
192 K de ROM.

ÉCRAN

80 caractères × 8 lignes
Résolution graphique de 640 × 80.
Caractères graphiques et internationaux intégrés.

INTERFACES

Cassette, lecteur de disquettes externe, sortie de modem, RS232, adaptateur d'alimentation CA, ports ROM RAM.

LANGAGES DISPONIBLES

BASIC Microsoft GW.

CLAVIER

De type machine à écrire standard avec 57 touches, dont 8 touches de fonction, des touches de commande du curseur, trois autres touches spéciales. Les valeurs des touches de fonction sont définies par logiciel et peuvent apparaître sur la dernière ligne de l'écran.

DOCUMENTATION

Les manuels fournis sont assez bons et comportent une information concernant l'installation du PC-5000 et l'utilisation du MS-DOS. Le manuel BASIC est trop technique.

FORCES

Le 5000 offre la plupart des fonctions d'un système de dimension normale. De plus, l'imprimante intégrée et les cartouches à bulles lui permettent de devancer ses rivaux.

FAIBLESSES

On ne peut déceler que de très rares faiblesses dans ce nouveau produit. Cependant, le 5000 est lourd pour ses dimensions et coûteux.

Rapport final

L'analyse des données est une tâche où les ordinateurs excellent. Même un système de gestion de stock très simple, comme le programme Dragon Data produit des rapports détaillés.

Sur le marché

Answare/Ère informatique/Loriciels

Les logiciels destinés à la petite gestion, encore appelée gestion familiale ou domestique, se multiplient.

Toute personne responsable d'une gestion de stock doit suivre l'ensemble des marchandises qu'elle doit gérer. Elle doit être en mesure de connaître à tout moment les niveaux et les mouvements de stock. Cela peut être fait de deux manières : soit au moyen d'interrogations à l'écran, soit à l'aide de rapports imprimés. Le menu d'interrogation du programme Dragon propose plusieurs options définissant le type de données disponibles et comment elles seront présentées.

Une des fonctions essentielles concerne les articles dont le mouvement est lent. Puisque la date de toutes les transactions est enregistrée lorsqu'elles sont entrées dans le système, le programme dispose déjà de toutes les informations dont il a besoin pour générer un rapport sur les articles à mouvement lent. Il n'a qu'à parcourir le fichier de l'historique des transactions et à comparer les dates. Puisque chaque activité a sa propre définition pour le terme « lent » (ce qui est lent pour l'une peut représenter un excellent rendement pour une autre), le rapport doit permettre à l'utilisateur de définir les conditions.

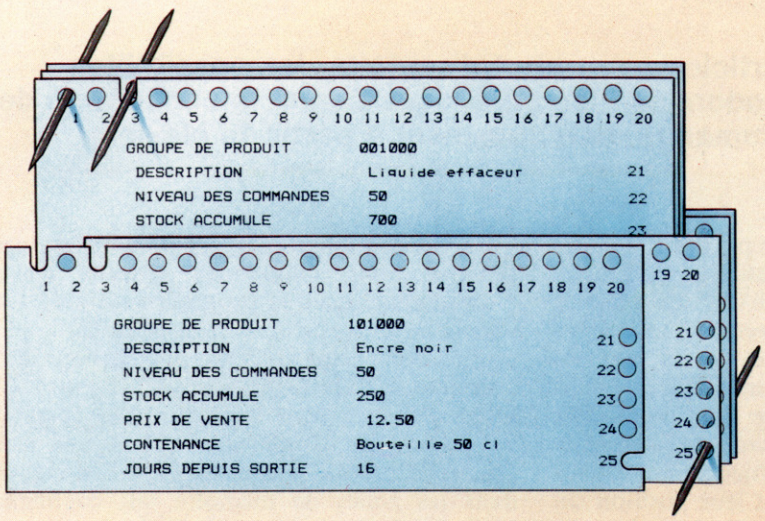
Cette exigence est correctement remplie par le système Dragon. En entrant une date dans la zone prévue à cet effet (comme 150484, pour 15 avril 1984), l'utilisateur donne automatiquement au système un marqueur pour effectuer la recherche. Tous les articles qui n'ont bénéficié d'aucune transaction de vente seront identifiés par le programme et imprimés. C'est très utile, puisque divers rapports de mouvement lent peuvent être produits en donnant simplement au programme une date différente. Seule contrainte imposée à l'utilisateur : la date doit se situer à l'intérieur de l'historique de transaction stocké sur ce fichier. Si aucun article à mouvement lent ne correspond à la date spécifiée, le programme Dragon produira le message suivant à l'écran : «AUCUN ARTICLE SOUS CE CRITÈRE DE SÉLECTION». Mais cette procédure a ses limites : ne sont extraits que les articles pour lesquels aucune transaction n'a été enregistrée.

Un système plus sophistiqué permettrait une plus grande souplesse. Il y a deux façons d'y arriver. Le système pourrait offrir, en plus de la date, un autre critère de sélection, comme par exemple une limite inférieure de transactions définissant quels articles font partie de la catégorie à mouvement lent. Le système pourrait aussi lire et comparer lui-même les mouvements sur toutes les lignes, et lister les 50 lignes les plus lentes, puis les 50 lignes suivantes les plus lentes, etc.

Affichage

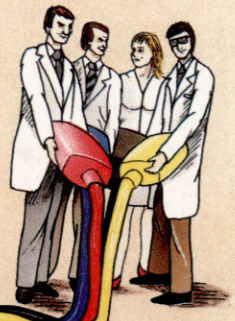
L'écran affiche chaque article séparément. Par conséquent, s'il existe un grand nombre d'articles à mouvement lent, le défilement de la liste pourrait constituer une perte de temps. Dans un tel cas, un rapport imprimé serait probablement préférable puisqu'il est plus rapide et plus facile à consulter. Le concepteur du système doit en tenir compte lors de l'écriture d'un programme de gestion de stock.

Sélection mécanique



Perforation
Le système de cartes à bordures perforées est une forme simple de bases de données. Les trous entourant la carte sont traités comme des chiffres binaires — un trou correspond à un *zéro*, une encoche correspond à un *un*. Ici, les cartes représentent le fichier de stock d'une entreprise : les trous 1 à 3 représentent le groupe de produits, les trous 4 à 6 composent le numéro de catalogue, les trous 1 à 6 forment donc ensemble le numéro de l'article; les trous 21 à 25 indiquent le nombre de jours écoulés depuis la dernière sortie de l'article, et le trou 25 correspond à 16 jours.

Faible rendement commercial
Le système de gestion de stock Dragon Data comporte un module de base de données qui génère une variété de rapports concernant l'état de l'inventaire. L'entreposage de stocks importants à mouvement lent est coûteux; l'entreprise recherche donc dans le groupe de produits 101 les articles non réclamés au cours des 16 derniers jours.



Kevin Jones

L'affichage du mouvement de stock propose également une information additionnelle. Il spécifie la valeur du stock, ce qui indique à l'utilisateur le montant du capital immobilisé dans un entrepôt. Il indique l'usage mensuel moyen, la quantité en stock et la date du dernier rapport. A partir de cette information, l'utilisateur pourra définir une politique commerciale différente; offrir un article avec une réduction importante, par exemple.

La notion de création de rapports fondée sur des critères de sélection comme la date est très importante lorsqu'on doit analyser des données de transaction. Évidemment, le système doit pouvoir lister les transactions concernant tous les articles. Mais il doit aussi illustrer l'évolution du stock entre des dates spécifiées.

Le programme Dragon accomplit cette tâche en demandant à l'utilisateur d'entrer les limites inférieure et supérieure de la valeur des articles et les dates limites (par exemple entre les dates 010184 et 010484).

En plus d'obtenir une liste des transactions concernant une gamme spécifiée d'articles, l'utilisateur peut aussi demander une liste par catégorie où les articles sont groupés selon le type de transaction, selon les chiffres de vente ou selon les ajustements de stock, etc. Le programme Dragon comporte un menu offrant trois options qui définissent le critère date pour la sélection (celles-ci sont : uniquement les transactions de la période en cours; toutes les transactions; les transactions effectuées à l'intérieur des dates spécifiées), et un autre sous-menu qui permet de fractionner la sélection en types de transaction.

Ce sous-menu inclut une option, «99 TOUS LES TYPES», qui permet d'obtenir une sortie imprimée de toutes les transactions correspondant aux paramètres de données spécifiés.

Le programme permet aux utilisateurs d'associer des articles de stock à des groupes de produits particuliers. Les utilisateurs peuvent revoir la composition des groupes de produits et leurs définitions. Ils désireront aussi savoir quels sont les stocks détenus dans des groupes de produits particuliers. Les rapports affichés à l'écran sont utiles mais ne constituent pas des relevés permanents. Afin d'offrir un relevé imprimé, la fonction rapport reproduit à de nombreux égards les fonctions d'interrogation offertes par ce programme.

Il y a de nombreuses raisons qui font du programme Dragon un système de gestion de stock plutôt limité, malgré ses fonctions analytiques évoluées. Le système est conçu comme un système de gestion de stock fermé. Il ne peut être relié à d'autres programmes de gestion associés qui pourraient utiliser l'information de ses fichiers. Il n'est pas possible de comparer le niveau du stock à celui des commandes. L'une des questions les plus importantes posées par les utilisateurs est la suivante : « Est-ce que je dispose d'un stock suffisant pour répondre à ces commandes? » Lorsque vous recevez plusieurs commandes comportant plusieurs articles différents, il est très difficile de suivre manuellement l'évolution des niveaux de stock.

Dans cette série d'articles nous nous sommes surtout intéressés à l'utilisation des ordinateurs domestiques pour la gestion de petites entreprises.

Malgré ces lacunes, les programmes de gestion sur ordinateur domestique peuvent grandement améliorer la comptabilité et l'organisation d'une petite entreprise.

Fais-moi un dessin!

Cette courte série d'articles a pour but d'exploiter les possibilités graphiques du Commodore 64 : les affichages écran, la conception de plans objets, et la commande du mouvement à partir du clavier.

L'une des caractéristiques les plus intéressantes du Commodore 64 est la possibilité qu'il offre d'écrire des jeux d'arcade en BASIC. La machine est en effet en mesure de créer des plans objets — des formes haute résolution qui peuvent être définies par l'utilisateur et manipulées facilement à l'écran. La mémoire du Commodore comporte des registres spéciaux qui commandent les attributs des plans objets, comme leurs couleurs, leurs dimensions et leur position sur l'écran. En écrivant (POKE) des nombres dans ces registres, le programmeur peut facilement commander l'action. Le jeu « Chasse aux sous-marins », que nous construirons ultérieurement à titre d'exemple, utilisera quatre des huit plans objets que peut créer la machine. Ils représenteront le navire, le sous-marin, les explosifs lancés du navire et une explosion.

Le jeu pourra ainsi être construit à l'intérieur d'articles successifs et chaque section du programme sera écrite sous la forme d'un court sous-programme qui sera appelé à partir du programme principal.

De nombreux lecteurs connaissent probablement le principe de ce jeu. Le joueur commande un navire qui poursuit les sous-marins. Ceux-ci traversent l'écran à des profondeurs et à des vitesses variables, et le navire lâche des charges explosives sur ces sous-marins. Le joueur marque des points chaque fois qu'un sous-mârin est touché, la valeur de chaque sous-marin étant définie en fonction de sa profondeur et de sa vitesse. Cependant, si le sous-marin s'échappe, sa valeur est soustraite du pointage du joueur. Le navire est commandé à partir du clavier, en utilisant les touches Z et X pour les mouvements horizontaux vers la gauche et vers la droite. Les charges sont lancées avec la touche M. Un chronomètre est affiché à l'écran; la durée du jeu est de trois minutes. A la fin de cette période, une autre manche est proposée au joueur et le programme affiche le meilleur score obtenu par le joueur depuis le début du programme.

Il est très important, lors de la conception d'un jeu d'action, de veiller à ce que la boucle principale du programme soit la plus courte possible. Le programme « Chasse aux sous-marins » utilise des sous-programmes pour exécuter les principales fonctions du jeu. Les seules fonctions contrôlées directement à partir du programme principal sont les suivantes : la mise à jour du chronomètre, l'acceptation d'une entrée au clavier, le déplacement du navire et le déplacement du sous-marin.

La conception de ce programme est assez générale pour s'appliquer à toute marque d'ordinateur, mais la programmation détaillée variera selon les caractéristiques de l'ordinateur utilisé. Dans cette partie du projet, nous allons examiner la routine qui crée un affichage.

Deux manières font apparaître des caractères sur l'écran du Commodore 64. L'une utilise l'instruction PRINT et l'autre écrit des nombres dans les zones de mémoire qui contiennent l'information concernant l'affichage.

L'écran du Commodore est composé de 25 lignes de 40 caractères. En d'autres termes, il existe 1 000 emplacements possibles sur l'écran pour placer des caractères. Chaque position de l'écran est associée à deux nombres. Le premier est un nombre de code écran qui indique à l'ordinateur quel caractère afficher à cette position. Le second est un code couleur qui indique de quelle couleur doit être le caractère affiché. Il existe deux blocs de mémoire et chacun comporte 1 000 emplacements : l'un de ces blocs contient l'information concernant le code écran et l'autre contient l'information concernant la couleur de chaque emplacement de l'écran. La zone qui renferme les codes écran est appelée la mémoire écran et occupe les emplacements 1024 à 2023. La mémoire couleur va de 55296 à 56295.

Chaque caractère a son propre code écran. Ces codes sont énumérés à la page 132 du guide utilisateur. Les codes des 16 couleurs du Commodore 64 sont les suivants :

0	noir	8	orange
1	blanc	9	brun
2	rouge	10	rouge clair
3	cyan	11	gris 1
4	violet	12	gris 2
5	vert	13	vert clair
6	bleu	14	bleu clair
7	jaune	15	gris 3

Deux autres emplacements présentent un intérêt particulier. L'emplacement 53280 commande la couleur de la bordure et 53281 commande la couleur de l'écran. Pour définir l'environnement de notre jeu, les six lignes du haut de l'écran seront bleu clair, tandis que le reste de l'écran et la bordure seront bleu foncé pour représenter la mer (à l'exception des deux lignes du bas qui représenteront le fond de la mer).

Pour régler la couleur de l'écran au bleu clair et la bordure au bleu foncé, la paire de commandes POKE suivante doit être utilisée :

```
POKE 53281,14 : POKE 53280,6
```

La septième ligne de l'écran commence à l'emplacement 1264. La deuxième ligne à partir du bas de l'écran commence à l'emplacement 1944. Nous pouvons colorer la mer en écrivant (POKE) le code écran correspondant à un espace inverse dans les adresses 1264 à 1943, et en écrivant 6 dans les adresses de mémoire couleur correspondantes. Il y a une méthode simple pour établir la liaison entre une adresse de mémoire couleur et l'adresse de mémoire écran correspondante : il suffit simplement d'ajouter 54272 à l'adresse de mémoire écran.

Le code écran désignant un caractère espace est 32. Un code écran de caractère inverse peut être obtenu en ajoutant 128 au code écran normal; ainsi, le code désignant un espace inverse est 32 + 128 = 160. Les lignes suivantes de programme utilisent une simple boucle FOR...NEXT :

```
FOR I=1264 TO 1943
POKE I,160:POKE I+54272,6
NEXT I
```

Le fond de la mer est formé de deux lignes de caractères en damier avec un code écran de 102 et de couleur brune. De nouveau, une simple boucle FOR... NEXT exécute cette tâche :

```
FOR I=1944 TO 2023
POKE I,102:POKE I+54272,9
NEXT I
```

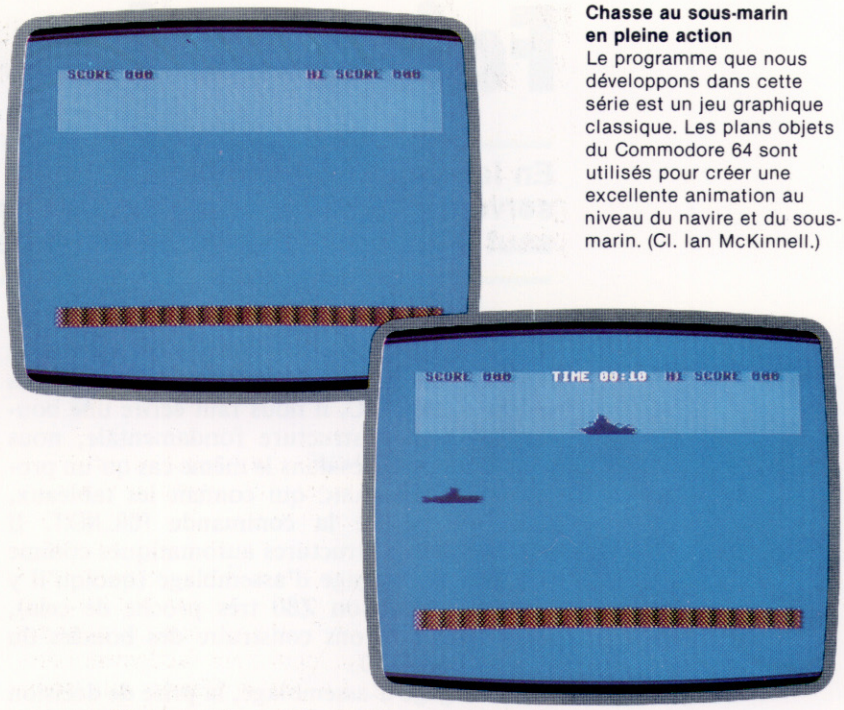
L'instruction PRINT représente également une méthode efficace pour produire des images. La couleur et la position du curseur peuvent être commandées avec une instruction PRINT en utilisant soit les caractères de commande spécifiques au Commodore ou encore les codes CHR$. Nous utiliserons la dernière méthode, puisque celle-ci nous permet d'obtenir des listages plus faciles :

CHR$(5)	blanc
CHR$(144)	noir
CHR$(147)	efface l'écran et place le curseur dans le coin supérieur gauche de l'écran
CHR$(19)	positionne le curseur dans le coin supérieur gauche
CHR$(17)	déplace le curseur d'une position vers le bas
CHR$(145)	déplace le curseur d'une position vers le haut
CHR$(157)	déplace le curseur d'une position vers la gauche
CHR$(29)	déplace le curseur d'une position vers la droite

A l'intérieur de notre routine de préparation d'écran, SCORE et HI SCORE doivent être imprimés sur la ligne du haut de l'écran. CHR$(19) garantira que le curseur se trouve au début de la ligne du haut de l'écran. La commande suivante imprime le pointage initial en noir :

```
PRINT CHR$(19);CHR$(144);«SCORE 000»
```

Le pointage maxi est positionné sur la ligne supérieure, mais du côté droit. La fonction SPC



Chasse au sous-marin en pleine action
Le programme que nous développons dans cette série est un jeu graphique classique. Les plans objets du Commodore 64 sont utilisés pour créer une excellente animation au niveau du navire et du sous-marin. (Cl. Ian McKinnell.)

permet d'insérer des espaces. La commande PRINT peut maintenant être modifiée pour inclure le pointage maxi :

```
PRINT CHR$(19);CHR$(144);
«SCORE 000»;SPC(6);HI SCORE 000»
```

La routine de préparation de l'écran formera un sous-programme qui commence à la ligne 1000 et qui est appelé par le programme principal. On y retrouvera une instruction POKE qui provoquera la répétition des touches lorsqu'elles seront maintenues enfoncées. Ce sous-programme peut être testé par les lignes suivantes :

```
10 GOSUB 1000:REM PRÉPARATION DE L'ÉCRAN
20 END

1000 REM **** PRÉPARATION DE L'ÉCRAN ****
1010 PRINT CHR$(147):REM EFFACER L'ÉCRAN
1020 :
1030 REM ** COLORATION DE LA MER **
1040 POKE 53281,14:POKE 53280,6
1050 FOR I=1264 TO 1943
1060 POKE I,160:POKE I+54272,6
1070 NEXT
1080 :
1090 REM ** FOND DE LA MER **
1100 FOR I=1944 TO 2023
1110 POKE I, 102:POKE I+54272,9
1120 NEXT
1130 POKE 650,128:REM RÉPÉTER LES TOUCHES
1140 :
1150 REM ** SCORE **
1160 PRINT CHR$(19):CHR$(144);
     «SCORE 000»;SPC(6);HI SCORE 000»
1170 RETURN
1180
1190
```

Il est préférable de sauvegarder cette routine sur bande ou sur disquette dès que vous avez terminé de la taper, avant de l'exécuter.

Faisons les comptes

En langage d'assemblage, boucles et branches conditionnelles servent à tester la condition de l'accumulateur, et les instructions de saut à changer le déroulement du contrôle dans le programme.

Avant de commencer à utiliser les différents modes d'adressage (en particulier les adresses indexées) de l'UC, il nous faut écrire une boucle. Sans cette structure fondamentale, nous sommes à peu près dans le même cas qu'un programmeur de BASIC qui connaît les tableaux, mais qui ignore la commande FOR...NEXT. Il n'existe pas de structures automatiques comme FOR...NEXT en langage d'assemblage (quoiqu'il y ait une instruction Z80 très proche de cela), mais nous pouvons construire des boucles du type IF...THEN GOTO...

En langage d'assemblage, la prise de décision est centrée sur les drapeaux du registre d'état du processeur (PSR). Ces drapeaux montrent les effets sur l'accumulateur de la dernière instruction exécutée, et on les appelle parfois *drapeaux de condition*. Tous ces drapeaux peuvent être utilisés pour prendre des décisions, mais nous n'aurons à en considérer que deux pour le moment : le drapeau de zéro (Z) et celui de retenue (C).

L'état de ces drapeaux peut être utilisé pour décider si le processeur exécutera la prochaine instruction du programme, ou s'il doit sauter à une autre instruction. La décision de continuer ou de sauter est donnée par le processeur qui change ou bien accepte l'adresse contenue dans son compteur de programme. Ce registre contient toujours l'adresse de la prochaine instruction en code machine à exécuter. Lorsque le processeur commence à exécuter une instruction, il charge l'opc dans l'instruction à partir de l'octet indiqué par l'adresse dans le compteur de programme. L'adresse dans le registre est incrémentée par le nombre d'octets de l'instruction, de sorte que le compteur de programme indique alors l'opc de l'instruction suivante. Si l'instruction en cours fait que le compteur de programme indique une adresse à un autre endroit du programme, alors un saut est effectué.

Sur le 6502, l'instruction BEQ a pour effet de modifier le compteur de programme si le drapeau de zéro est mis. BCS est l'instruction équivalente si le drapeau de retenue est mis. Sur le Z80, ces instructions sont JR Z et JR C respectivement. Ces quatre opc sont appelés *instructions de branchement* parce qu'elles représentent un point de branchement dans le déroulement du contrôle de programme. Leur opérande est un nombre à un octet, qui est additionné à l'adresse dans le compteur de programme pour fournir une nouvelle adresse. Considérons ce qui se passe lorsque le programme suivant est exécuté :

ORG $5E00			
6502		Z80	
5E00	ADC #$34	ADC	A,$34
5E02	BEQ $03	JR	Z,$03
5E04	STA $5E20	LD	($5E20),A
5E07	RTS	RET	

Si l'instruction ADC en $5E00 produit un zéro dans l'accumulateur (ce qui est improbable, mais, comme nous le verrons plus loin, possible), alors les instructions BEQ et JR Z en $5E02 auront pour effet d'ajouter $03 au contenu du compteur de programme. L'instruction suivante à exécuter sera donc l'instruction de retour en $5E07, ce qui a pour effet de sauter l'instruction en $5E04.

A première vue, cela peut sembler faux : si l'instruction en $5E02 fait que $03 est additionné au compteur de programme, l'adresse qui y est stockée ne devrait-elle pas devenir $5E05? Mais il est important de se souvenir que le compteur de programme indique toujours la *prochaine* instruction à exécuter et non l'instruction en cours. Ainsi, quand commence l'exécution de l'instruction en $5E02, le compteur de programme contiendra l'adresse $5E04 — l'emplacement de l'instruction suivante. Si l'on additionne $03 à $5E04, le résultat sera $5E07, adresse de la prochaine instruction.

Remarquons que le processeur n'est pas capable de vérifier si les adresses indiquées sont correctes. Si, par inadvertance, nous changeons le déplacement dans l'instruction en $02, alors le compteur de programme sera incrémenté (si l'accumulateur contient zéro) de $02, et le processeur considérera que $5E06 est l'adresse de l'opc de la prochaine instruction. Dans le programme correct, $5E06 contient la valeur $5E, qui est l'octet hi de l'opérande de l'instruction en $5E04. Néanmoins, le processeur ne peut juger si l'instruction est bonne ou non. Pour lui, $5E est un opc valable et l'exécution continuera, en prenant les octets suivant $5E06 comme opérandes de l'instruction.

En langage d'assemblage, cependant, le calcul des sauts ne pose pas de problèmes parce que le programme assembleur peut le faire à notre place. C'est ainsi qu'au lieu de fournir un déplacement hex en tant qu'opérande de l'instruction de branchement, nous donnons l'adresse symbolique de l'instruction à laquelle

il faut sauter. Il en résulte que le programme en langage d'assemblage est bien plus facile à suivre. L'assembleur décode l'adresse symbolique en adresse absolue, calcule le déplacement nécessaire pour avoir l'adresse, et transcrit ce déplacement en instruction code machine. L'adresse symbolique est appelée *label* (analogue au numéro de ligne d'un programme BASIC).

Examinons de plus près la façon dont on utilise les labels. Un label est une chaîne alphanumérique écrite au début d'une instruction en langage d'assemblage. Il est traité par le programme assembleur comme un symbole à deux octets remplaçant l'adresse du premier octet de l'instruction. Nous pouvons donc réécrire le programme précédent comme ceci :

ORG $5E00		
	6502	Z80
5E00	ADC #$34	ADC A,$34
5E02	BEQ EXIT	JR Z,EXIT
5E04	STA $5E20	LD ($5E20),A
5E07 EXIT	RTS	RET

L'instruction en $5E02 peut maintenant être lue : « SI la valeur de l'accumulateur est zéro ALORS ALLER à l'adresse représentée par le label EXIT. » Ce qui est bien plus lisible que la première version et diminue considérablement les risques de se tromper dans le branchement. Nous pouvons à présent utiliser des labels et les instructions de branchement pour créer une boucle :

ORG $5E00		
	6502	Z80
5E00 START	ADC #$34	ADC A,$34
5E02	BNE START	JR NZ,EXIT
5E04	STA $5E20	LD ($5E20),A
5E07 EXIT	RTS	RET

Notez ici l'utilisation du nouveau label, START, ainsi que des nouvelles instructions de branchement : BNE qui signifie « brancher si l'accumulateur n'est pas (not) égal à zéro ». Considérons l'effet de ce code. Le programme commencera par additionner $34 à l'accumulateur. Si le résultat n'est pas égal à zéro, le programme se rebranche en $5E00 — adresse représentée par le label START. $34 sera à nouveau additionné à l'accumulateur, et du résultat dépendra éventuellement un autre branchement. Cette « boucle » se poursuit jusqu'à ce que la condition de branchement soit remplie. Si le contenu de l'accumulateur est nul après une instruction ADC, alors le branchement en $5E02 n'aura pas lieu, et l'instruction en $5E04 sera exécutée ensuite.

C'est exactement comme une boucle IF... THEN GOTO... en BASIC, à ceci près qu'il n'est pas difficile de voir que l'accumulateur ne pourrait jamais s'annuler. Après tout, il est incrémenté de $34 chaque fois que la boucle est exécutée! Comment alors pourrait-il redevenir nul? La réponse provient de ce que l'accumulateur est un registre à un seul octet, et si le résultat de l'addition est un nombre à deux octets, alors le drapeau de retenue du PSR sera mis, et l'accumulateur contiendra l'octet lo du résultat. Si l'accumulateur contient $CC, par exemple, alors en additionnant $34 on obtiendra le nombre à deux octets $0100. Le drapeau de retenue sera mis, et l'accumulateur contiendra l'octet lo de ce résultat — $00. Ainsi, le contenu de l'accumulateur sera nul, et le drapeau de zéro sera donc mis.

Compte tenu de ce résultat, nous pourrions réécrire le programme.

ORG $5E00		
	6502	Z80
5E00 START	ADC #$34	ADC A,$34
5E02	BCC START	JR NC,START
5E04	STA $5E20	LD ($5E20),A
5E07 EXIT	RTS	RET

Dans cette version, l'instruction en $5E02 devient : « si le drapeau de retenue est à zéro, se brancher à START. » Dès que le résultat de l'addition de $34 à l'accumulateur est supérieur à $FF, alors le drapeau de retenue est mis, et le branchement à l'adresse de START n'a pas lieu.

Compteur de boucle

Le branchement selon les conditions actuelles du drapeau de retenue ou du drapeau de zéro est une possibilité assez limitée, mais elle permet toute une série de décisions, comme nous allons le voir bientôt. Ce qui manque en fait à notre répertoire est la possibilité de tenir un *compteur de boucle*. Nous pourrions avoir besoin de compter le nombre de fois qu'une boucle est parcourue avant que la condition de sortie ne soit remplie, ou de provoquer la sortie d'une boucle après un nombre donné d'itérations :

6502		
0000	ORG	$5DFD
5DFD	LDX	#$00
5DFF START	INX	
5E00	ADC	#$34
5E02	BCC	START
5E04	STX	$5E20
5E07 EXIT	RTS	

Z80		
0000	ORG	$5DFA
5DFA	LD	IX,$0000
5DFE START	INC	IX
5E00	ADC	A,$34
5E02	JR	NC,START
5E04	LD	($5E20),IX
5E08 EXIT	RET	

La nouvelle structure a impliqué plusieurs modifications dans le programme. Primo, les instructions insérées au début du programme requièrent une nouvelle adresse ORG. Ces instructions ont à peu près les mêmes effets sur les processeurs 6502 et Z80, mais leurs longueurs sont différentes, de sorte que les adresses ne sont plus les mêmes dans les deux versions du programme. Secundo, de nouvelles versions des instructions de chargement (LDX, LD IX) et de stockage (STX, LD(), IX) ont été employées pour mettre

une valeur initiale de $00 dans le registre d'index de l'UC. Le registre 6502 X est un registre à un octet, mais le IX du Z80 a deux octets. Les registres d'index ont des fonctions spéciales, mais ce sont essentiellement des RAM d'UC tout comme l'accumulateur, et ici nous les utilisons comme des accumulateurs supplémentaires servant à compter les boucles. Au moment où l'on sort de la boucle, le contenu du registre 6502 X est stocké en $5E20. Dans la version Z80, l'octet lo du registre IX (à deux octets) est stocké en $5E20 et l'octet hi en $5E21.

Tertio, une toute nouvelle instruction a remplacé ADC au début (START) de la boucle : INX et INC IX.sont toutes deux des instructions d'incrémentation, qui augmentent (ou diminuent) de $01 le contenu du registre d'index. Cela actualise la valeur du compteur de boucle à chaque tour.

Voici comment nous pouvons comprendre le programme : « mettre le compteur de boucle à zéro, commencer la boucle en incrémentant le compteur, ajouter $34 à l'accumulateur, et se rebrancher au début de la boucle si le drapeau de retenue est à zéro, sinon stocker le contenu du compteur de boucle en $5E20. »

6502		
0000	ORG	$5DFA
5DFA	LDX	#$00
5DFC START	STA	$5E22,X
5DFF	INX	
5E00	ADC	#$34
5E02	BCC	START
5E04	STX	$5E20
5E07 EXIT	RTS	

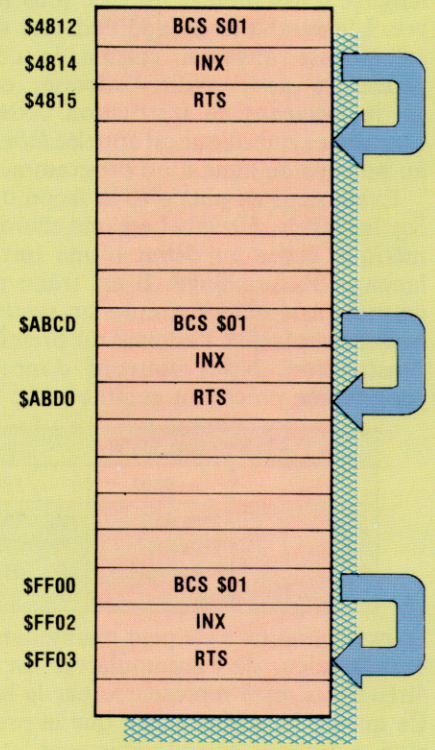
Z80		
0000	ORG	$5DF7
5DF7	LD	IX,$5E00
5DFB START	LD	(IX+$22),A
5DFE	INC	IX
5E00	ADC	A,$34
5E02	JR	NC,START
5E04	LD	($5E20),IX
5E08 EXIT	RET	

Les versions 6502 et Z80 ont le même effet : créer en $5E22 un tableau pour stocker les valeurs successives de l'accumulateur à mesure que le programme est exécuté, et éventuellement stocker en $5E20 la valeur finale du compteur de boucle, qui est aussi le nombre d'octets dans le tableau à partir de $5E22.

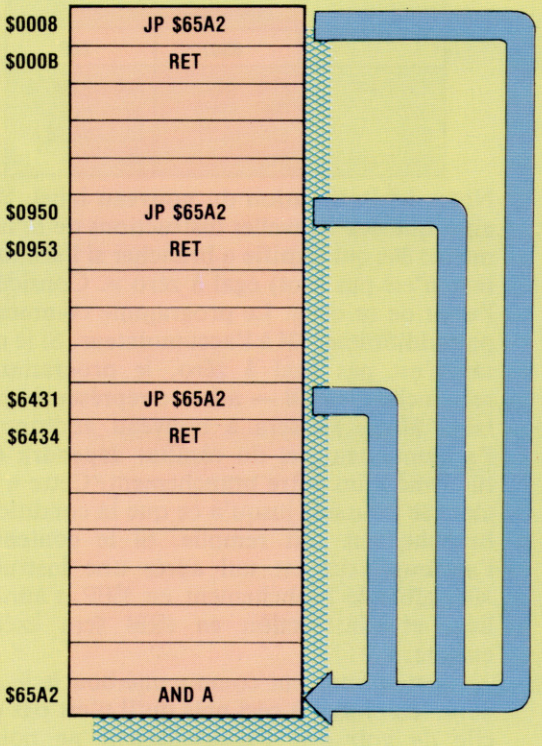
La version 6502 réalise cela à l'aide de l'instruction STA $5E22,X qui signifie « additionner le contenu du registre X à l'adresse de base, $5E22, puis stocker le contenu de l'accumulateur à l'adresse ainsi formée ».

L'instruction STA est ici en mode indexé direct absolu : c'est-à-dire que le registre X est utilisé comme un index pour modifier l'adresse de base, $5E22. Puisque le registre X est initialisé à $00, et en conséquence incrémenté à chaque itération, la valeur de départ de l'accumulateur sera stockée en $5E22, la valeur suivante en $5E23, et ainsi de suite. Après la sortie de la boucle, STX stockera la valeur finale du compteur de boucle à l'adresse $5E20.

Sauts relatifs



Sauts absolus



La version Z80 utilise le registre IX comme un pointeur pour indiquer l'adresse de stockage en cours, tout en utilisant l'octet lo de IX comme compteur de boucle. L'instruction LD IX,$5E00 met l'adresse de base, $5E00, dans le registre IX, de sorte que l'octet lo de IX contient $00. L'instruction un peu particulière LD (IX+$22),A signifie : « additionner l'adresse contenue dans IX à $22, et stocker le contenu de l'accumulateur à l'adresse ainsi obtenue. » Puisque IX est initialisé à $5E00, puis incrémenté à chaque itération de la boucle, la valeur de départ de l'accumulateur sera stockée en $5E22, la suivante en $5E23, et ainsi de suite. Pendant ce temps, l'octet lo de IX enregistre le nombre d'itérations de boucle, et est finalement stocké en $5E20 à la fin de la boucle. L'instruction LD (IX+$22),A est en mode d'adressage indexé indirect absolu, qui est bien plus compliqué que la version 6502 mais beaucoup plus puissant.

Voilà ce que l'on peut dire des boucles et des structures de tableaux en langage d'assemblage. Toutes deux sont extrêmement utiles.

Sauts relatifs
La plupart des instructions de branchement, telles que BCS (branchement si le drapeau de retenue est mis), JR NZ (branchement si l'accumulateur n'est pas à zéro), fonctionnent selon la condition du PSR, et utilisent le mode de saut relatif pour réorienter le déroulement du contrôle dans le programme.
Dans cet exemple, l'instruction BCS $01 provoque toujours un saut relatif d'un octet en avant (si toutefois l'état du drapeau de retenue implique un saut), quelle que soit l'adresse où réside le code machine. Ici, BCS$01 est toujours suivi de l'instruction INX (à un octet); lorsque le drapeau de retenue est mis, BCS aura donc pour effet de sauter l'instruction INX.

Sauts absolus
Dans cet exemple, l'instruction JP $65A2 provoque un saut inconditionnel chaque fois qu'elle est rencontrée. Son effet est de re-diriger l'exécution du programme à l'adresse qui forme son opérande — $65A2 ici. Aucun test n'est effectué, et l'adresse de l'instruction au moment de l'exécution n'a pas d'importance.
Les deux modes de sauts ont des avantages et des inconvénients, mais le critère de choix le plus important entre saut relatif ou absolu est la possibilité de réallocation : il est très courant, en programmation en langage d'assemblage, d'écrire une routine et de l'assembler à une adresse ORG, puis de la ré-utiliser sous la même forme mais avec une autre valeur de ORG. Si tous les sauts dans une routine sont relatifs, alors le fait de changer les adresses des instructions n'aura pas de conséquence, et le programme se déroulera comme prévu; si l'un des sauts est absolu lorsque la routine est assemblée en un différent ORG, les sauts renverront toujours à l'adresse spécifiée, ce qui peut ne plus avoir de signification pour la routine.

Exercices

Il y a dans ce numéro un grand nombre de points importants, voire intrigants, et seule l'expérience de l'emploi des nouveaux modes d'adressage et instructions vous les fera pleinement comprendre.

Utilisez le programme assembleur CHAMP et sauvegardez les différents fragments de programme de ce numéro. Lorsque vous exécuterez un fragment, utilisez le mode debug pour examiner les adresses mémoire qui pourraient être affectées. Il est toujours bon d'initialiser ces adresses par une constante reconnaissable — $FF, par exemple — avant l'exécution, afin de savoir si la mémoire a été affectée par le programme. Vous pouvez utiliser la commande debug de modification.

N'oubliez pas que, comme toujours, les adresses données dans le programme ne sont que des exemples, et que vous devrez choisir des adresses convenant à votre machine.

Chargement et sauvegarde de CHAMP

Par commodité et sécurité, vous devrez copier CHAMP sur une autre bande, puis retirer les plaquettes de protection contre l'écriture, de l'original et de la copie. Dans les instructions suivantes, LOAD se réfère à la bande CHAMP, et SAVE à la bande copie.

BBC Modèle B
1. LOAD«CHAMP»
2. SAVE«CHAMP»:BRUN:passer au BASIC
3. *SAVE«CHAMP M/C» 1000,4600

Commodore 64
1. LOAD«CHAMP»
2. SAVE«CHAMP»:RUN:entrer mode <debug>
3. Taper [w] [ret], suivi de [s] pour SAVE
4. Adresse de début 1000;adresse de fin 4600 nom de fichier «CHAMP M/C»

Spectrum
1. LOAD«CHAMP»
2. Passer au BASIC:SAVE«CHAMP»LINE 1
3. SAVE«CHAMP M/C» CODE 27000,9231

Les instructions de branchement conditionnel, comme nous l'avons vu, dépendent du contenu du PSR. Une des raisons pour ajouter l'option d'affichage binaire au programme moniteur était de vous permettre d'inspecter le contenu du PSR avant et après l'exécution d'une instruction, et d'observer les changements de drapeaux. Il n'y a pas une seule instruction en langage d'assemblage 6502 ou Z80 pour stocker le contenu du PSR, nous devons donc faire ainsi :

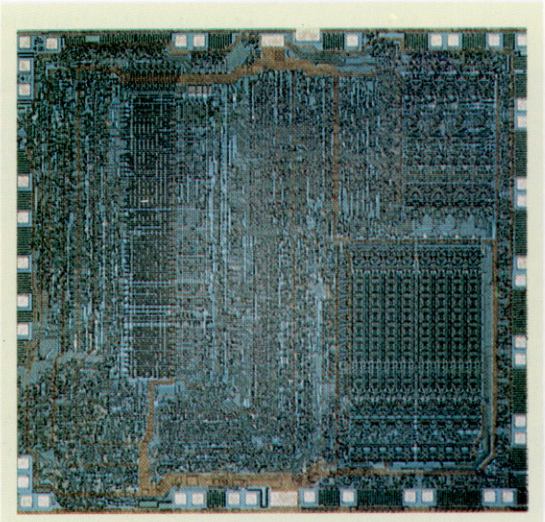
Z80	
3E00 F5	PUSH AF
3E01 F5	PUSH AF
3E02 E1	POP HL
3E03 22 lo hi	LD (STORE1),HL
3E06 F1	POP AF
6502	
3E00 48	PHA
3E01 08	PHP
3E02 48	PHA
3E03 08	PHP
3E04 68	PLA
3E05 8D lo hi	STA STORE1
3E08 68	PLA
3E09 8D lo' hi'	STA (1+STORE1)
3E0C 28	PLP
3E0D 68	PLA

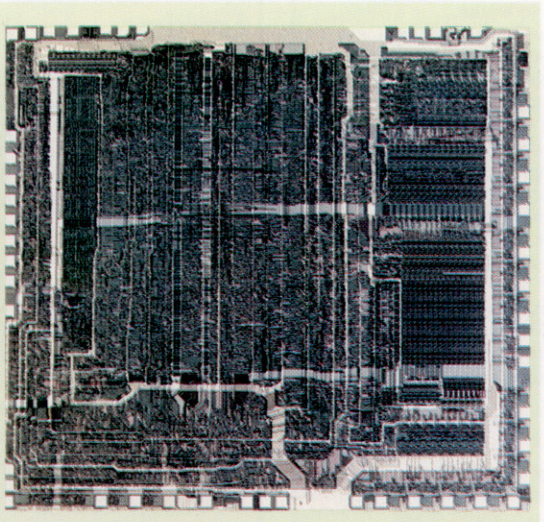
Cette séquence d'instructions fera que le contenu actuel du PSR sera stocké dans l'octet adressé par STORE1 (une adresse appropriée à votre machine), tandis que le contenu de l'accumulateur sera stocké en (1+STORE1). Pour utiliser ces instructions, insérez-les simplement en bloc avant et après le programme dont vous voulez observer l'effet. Vous devez toutefois vous rappeler qu'il faut additionner deux à la valeur de STORE1 chaque fois que vous insérez ce bloc. Lorsque vous aurez exécuté le programme, vous pourrez utiliser le moniteur pour afficher la section de mémoire où vous avez stocké les différents contenus du PSR et de l'accumulateur.

Il vous est possible de traiter ce bloc comme un sous-programme au lieu de l'entrer à chaque fois qu'il est nécessaire. Le langage d'assemblage possède un équivalent du BASIC GOSUB, mais son usage ici compliquerait les choses car il utilise la pile, qui interférerait avec l'utilisation du bloc dans la même liste (PLA, PUSH, PHP, etc. sont toutes des manipulations de pile, que nous expliciterons ultérieurement). Vous pouvez noter la différence de longueur entre les codes Z80 et 6502 : elle est due aux registres à deux octets du Z80.

Dresseurs de puces

Lorsqu'en 1977 Zilog Inc. lança sur le marché le microprocesseur Z80, peu de gens eurent l'impression d'assister à une révolution. Pourtant, le Z80 est entré dans presque tous les foyers.





Sauts de puce
Ces deux microphotographies illustrent bien la complexité des circuits intégrés contenus dans un microprocesseur. L'image de gauche est un très fort agrandissement du Z80, dont le succès fut énorme; celle de droite représente un Z8000, de création plus récente. Étant un 16-bits, le Z8000 possède plus de connections (les petits carrés situés sur les bords), et son schéma d'organisation est bien plus dense.

L'histoire de Zilog commence au début des années soixante-dix. Frederico Faggin et Masatoshi Shima, deux employés d'Intel — un producteur de circuits intégrés — décidèrent de fonder leur propre compagnie. Ils avaient participé à la mise au point du 8080A (qu'on a décrit comme étant « le premier ordinateur contenu dans une puce »). Tirant parti de cette expérience, ils entreprirent d'aller plus loin. Le 8080 connaissait déjà un grand succès auprès des concepteurs comme des amateurs passionnés et les deux hommes, tout naturellement, cherchèrent à créer une puce qui serait compatible avec lui. Cela permettrait, entre autres, d'accéder aux programmes déjà écrits pour lui. Grâce à leur expérience de première main du 8080, ils parvinrent à accroître le jeu d'instructions (les commandes en langage machine contenues dans le microprocesseur), grâce à l'introduction de registres supplémentaires, de codes opératoires à deux octets, etc. Le résultat final — le Z80 — représentait un progrès considérable.

Cette révolution au niveau des matériels s'accompagna par chance d'un bouleversement analogue au niveau des logiciels. En 1972, Gary Kildall et John Torode avaient écrit un programme appelé Control Program/Monitor — en abrégé CP/M — qui permettait à un microprocesseur de gérer les systèmes de disquettes, qui venaient de faire leur apparition. Kildall était lui-même consultant auprès d'Intel, et CP/M était donc prévu pour pouvoir tourner sur 8080 et 8085. Ce logiciel devint très vite le système d'exploitation de disquettes dominant sur le marché des micro-ordinateurs; Zilog se retrouvait donc dans des conditions idéales pour tirer parti de cet engouement, puisque le Z80 était compatible avec le 8080.

Cette réussite triomphale fut pourtant à l'origine des difficultés ultérieures. Si le Z80 se vend toujours aussi bien — la firme produit un million d'unités par mois —, le passage au stade des puces de 16 bits s'est révélé très épineux.

Le Z8000 fut la première tentative de Zilog en ce domaine. Si l'on y voit généralement un microprocesseur très puissant, pourvu d'un vaste jeu d'instructions et de nombreux registres, il est malheureusement très difficile à programmer. De surcroît, il n'est pas compatible avec le Z80, et il ne peut donc pas exploiter les innombrables programmes écrits pour ce dernier. Les constructeurs désirant mettre au point un ordinateur 16 bits avaient par conséquent tendance à rechercher une puce moins exigeante. Bien que compatible avec le Z80000 (ou Z80K) à 32 bits — mais ce dernier n'est pas encore disponible commercialement — le Z8000 ne connut donc qu'un succès limité. Les concepteurs de Zilog retournèrent à leurs planches à dessin, et la compagnie doit bientôt lancer le Z800, un 16 bits qui sera cette fois compatible avec le vénérable Z80. Celui-ci vient d'être choisi comme unité centrale des micros répondant aux normes MSX adoptées par de nombreux constructeurs japonais; et Commodore annonce une nouvelle série autour du Z8000.

Frank de Weeger, le président de Zilog

PROGRAMME N° 20

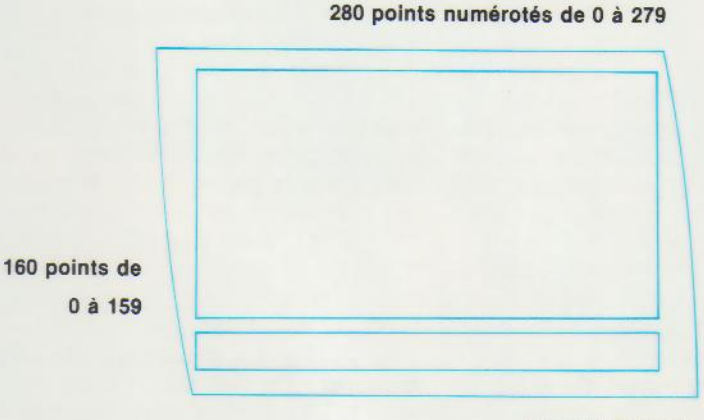
GRAPHIQUE HAUTE RÉSOLUTION

On a vu jusqu'à présent l'instruction GR permettant de travailler en BASSE RÉSOLUTION GRAPHIQUE. Sur Apple, on a une autre possibilité : travailler en HAUTE RÉSOLUTION GRAPHIQUE en HGR.

Instruction HGR : passage en mode graphique HAUTE RÉSOLUTION.
Cette instruction fait passer en mode graphique HAUTE RÉSOLUTION. L'écran est divisé en 280 × 160 points numérotés en abscisse de 0 à 279 et en ordonnée de 0 à 159.
Ces points peuvent être allumés par la commande :

HPLOT X, Y,

X, Y étant les coordonnées du point.
HCOLOR : ... permet de choisir une couleur.



HCOLOR = couleur : choix de la couleur

0	NOIR	4	NOIR
1	VIOLET	5	BLEU
2	VERT	6	ROUGE
3	BLANC	7	BLANC

HPLOT X, Y : allume le point de coordonnée X, Y dont la couleur a été déterminée par HCOLOR = COULEUR.

Dessin d'une ligne
L'instruction HPLOT XD, YD to XA, YA.

XD, YD coordonnée de départ
XA, YA coordonnée d'arrivée.

Un petit exemple : tracé d'une droite.

```
10   REM   PASSAGE EN MODE HAUTE RE
      SOLUTION GRAPHIQUE
20   HGR
30   REM   CHOIX DE LA COULEUR
40   HCOLOR= 5
45   REM   DESSIN DE LA DROITE
50   HPLOT 5,130 TO 60,70
60   REM   TAPEZ UNE TOUCHE
70   INPUT Z$
80   REM   RETOUR AU MODE TEXT
90   TEXT
```

Traçons des droites multiples.
L'instruction HPLOT A1, B1 to A2, B2 TO A3, B3 TO A4, B4.
Les tracés de cette suite de droites se fait en spécifiant les coordonnées des différents points de départ et d'arrivée.

Exemple : traçons une figure géométrique.

```
10   REM   PASSAGE EN HGR
20   HGR
30   REM   CHOIX DE LA COULEUR
40   HCOLOR= 3
50   REM   DESSIN D'UNE FIGURE GEOM
      ETRIQUE
60   HPLOT 5,5 TO 5,80 TO 80,80 TO
      80,5 TO 5,5
70   REM   TAPEZ UNE TOUCHE POUR RE
      TOUR AU MODE TEXT
80   GET Z$
90   REM   RETOUR EN TEXT
100  TEXT
```

Autre exemple : dessin d'une suite de rectangles de dimensions différentes. (Voir organigramme ci-dessous.)

```
10   REM   PASSAGE EN HGR
20   HGR
30   REM   CHOIX DE COULEUR
40   HCOLOR= 12
50  H = 8:L = 12
60  A1 = 15:B1 = 15
70  K = 1.25
80   FOR I = I TO 10
90   HPLOT A1,B1 TO A1,B1 + H TO A
      1 + L,B1 + H TO A1 + L,B1 TO
      A1,B1
100 A1 = A1 * K:B1 + B1 * K:L = L
      * K:H + H * K
110  NEXT I
120  GET Z$
130  TEXT
```

ORGANIGRAMME